SINGLE-TARGET TRACKING OF ARBITRARY OBJECTS USING MULTI-LAYERED FEATURES AND CONTEXTUAL INFORMATION

by

JINGJING XIAO

A thesis submitted to
The University of Birmingham
for the degree of
DOCTOR OF PHILOSOPHY

Department of Electronic, Electrical and Systems Engineering
College of Engineering and Physical Sciences
The University of Birmingham
March 2016

ABSTRACT

This thesis investigated single-target tracking of arbitrary objects. Tracking is a difficult problem due to a variety of challenges such as significant deformations of the target, occlusions, illumination variations, background clutter and camouflage. To achieve robust tracking performance under these severe conditions, this thesis proposed firstly a novel RGB single-target tracker which models the target with multi-layered features and contextual information. The proposed algorithm was tested on two different tracking benchmarks, i.e., VTB and VOT, where it demonstrated significantly more robust performance than other state-of-the-art RGB trackers. Proposed secondly was an extension of the designed RGB tracker to handle RGB-D images using both temporal and spatial constraints to exploit depth information more robustly. For evaluation, the thesis introduced a new RGB-D benchmark dataset with per-frame annotated attributes and extensive bias analysis, on which the proposed tracker achieved the best results. Proposed thirdly was a new tracking approach to handle camouflage problems in highly cluttered scenes exploiting global dynamic constraints from the context. To evaluate the tracker, a benchmark dataset was augmented with a new set of clutter sub-attributes. Using this dataset, it was demonstrated that the proposed method outperforms other state-of-the-art single target trackers on highly cluttered scenes.

Declarations

The author confirms that the work presented in this thesis is her own work. The contents of this thesis are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other universities. This thesis is author's own work and appropriate credit has been given where the references are provided to others' work.

Jingjing Xiao

March 2016

Acknowledgements

Life can be especially hard for those academics. I very often feel like a little coal miner, who has to work in this dark world while still trying to dig out a little “treasure” under the ground of science. Luckily, there are so many great people coming to my life as bright lamps lightening this world. Things that has happened in the past and present will inevitably help define my future and mould me into someone more complete.

As an international student, the first one I should thank is my country — China, who gives me everything. All the imperfections there become my essential motivation of hard working. During three and half years PhD study, my knowledge about research is like a canvas furnished by the unreserved dedication from my supervisors, Ales Leonardis, Rustam Stolkin, Robert Stone and Mourad Oussalah. There are so many things that I still want to learn, discover and share with them, which leads to the mixed feelings about my graduation today. I always expected that one day I could wear the graduation outfit while I was also so scared of the day coming and I have to leave them. However, if we want the blue sky, we have to fly. No matter how far I travel, I know there is always an inner power supporting me to overcome every difficult period.

Special thanks also go to all my family members, friends, and everyone that I met in my life, as they taught me how to percept and love this world. Their unconditional help and love contributes all of me. From Chongqing (China) to Changsha (China), from Changsha (China) to Birmingham (U.K.), I am such a lucky girl that has a chance to experience this colourful world. Now, it is my time to give back and paint the world.

CONTENTS

ABBREVIATIONS

LBP . . . . . Local binary pattern

HOG . . . . . Histogram of gradients

SIFT . . . . . Scale invariant feature transform

CNN . . . . . Convolutional neural networks

LR . . . . . Logistic regression

SVM . . . . . Support vector machines

SLIC . . . . . Simple linear iterative clustering

AUC . . . . . Area under curve

VTB . . . . . Visual tracking benchmark [2]

VOT . . . . . Visual object tracking challenges [3]

ALOV . . . . . Amsterdam library of ordinary videos for tracking [4]

PTB . . . . . Princeton tracking benchmark [5]

LGT . . . . . Robust Visual Tracking using an Adaptive Coupled-layer Visual Model [6]

SCM . . . . . Robust object tracking via sparsity based collaborative model [7]

KCF . . . . . High-speed tracking with kernelised correlation filters [9]

SD-KCF . . . . . Depth scaling kernelised correlation filters [10]

OAPF . . . . . Occlusion aware particle filter tracker [11]

TINF . . . . . Target identity-aware network flow [12]

SMOT . . . . . Tracking multiple targets with similar appearance [13]

CHAPTER 1

INTRODUCTION

1.1 Motivation

Tracking is a fundamental task in computer vision with numerous applications, e.g. motion analysis, event detection, scene understanding, human-computer interaction, augmented reality and robot navigation. This has motivated extensive research in recent years to improve the efficiency of automatic target tracking from video sequences [14]. Despite the publication of numerous tracking algorithms over the last three to four decades [2, 3], visual object tracking remains an open and active research area with several remaining unsolved problems, e.g. significant target deformation, field of view change, illumination variation, background clutter, camouflage and occlusion, which motivates the work of tackling the single-target tracking problems for arbitrary objects. In the following three subsections, we will present detailed motivations in three demanding scenarios: tracking arbitrary objects in RGB images; extension to tracking in RGB-D images; and tracking in scenes where many other objects or image regions share similar appearance with the target.

1.1.1. A single target tracker in RGB images.

After several decades of visual tracking research, even the most sophisticated trackers are still prone to failure in challenging scenarios, including clutter and camouflage in one or more feature modalities, rapid and erratic target motion, occlusions, and targets which change their shape and appearance over time [1], shown in figure 1.1. These severe tracking conditions

predominantly result in failures in three fundamental aspects of the tracking algorithms [15]: 1) the algorithm's model of the appearance of the target; 2) the mechanisms for matching model parts to image regions; 3) the mechanism for automatic model adaptation, when the target changes its appearance over time. Therefore, to tackle such problems in RGB images, it is important to design a tracker while keeping these three key aspects in mind.

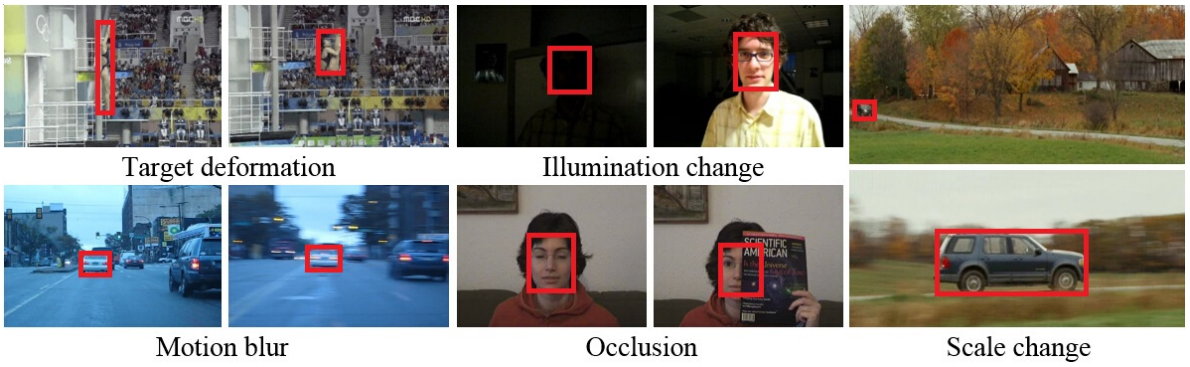

Figure 1.1: Tracking challenges including target significant deformation, illumination change, motion blur, scale change and occlusion.

1.1.2. A single target tracker in RGB-D images.

To further improve tracking performance, recent work has tried to incorporate additional features [16, 17]. Due to the proliferation of the affordable depth sensors, like Microsoft Kinect, Asus Xtion and PrimeSense etc., depth information is attracting growing interest [5]. However, while RGB features can be comparatively invariant to target motion, depth information can vary rapidly during target motion. For this reason, many state-of-the-art RGB-D methods [18, 19, 11] rely mainly on RGB data for target tracking, and reserve depth information primarily for occlusion reasoning. However, in such methods, rapid target motion towards the camera can be easily mistaken for an occlusion, leading to tracking failure, shown in figure 1.2. Therefore, how to exploit the depth information in a robust way is an essential but unsolved problem for a RGB-D tracker.

1.1.3. A single target tracker with scene distractors

Using previously mentioned tracking components (target model, matching method and model adaptation), a properly designed single target tracker can overcome many conventional challenges, e.g. significant deformation, illumination change, fast movement etc. However, to

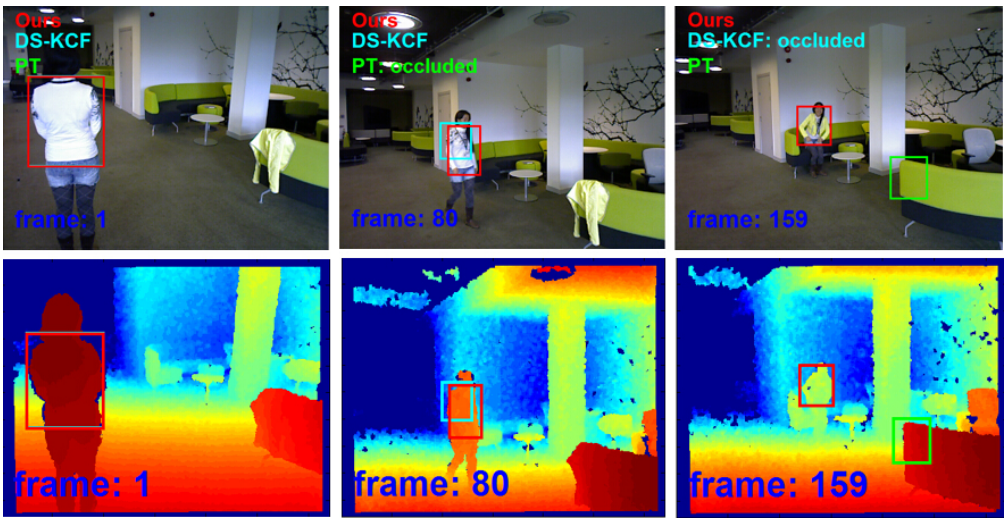

Figure 1.2: RGB-D tracking of target with large depth variation. Red, blue, green bounding boxes represent our proposed tracker and the state-of-the-art trackers DS-KCF [10] and PT [20] respectively. In frame 80, following rapid target depth change, PT [20] falsely reports "occluded" status (bounding box disappears). In frame 159, DS-KCF [10] falsely reports "occluded" status, while PT [20] has completely failed and drifted onto background clutter because of rapid depth change. It is essential to exploit the depth information in a more robust way.

robustly handle occlusion, ambiguity and camouflage problems in highly cluttered scenes, the tracker should "interact" with contextual information to help differentiate the target from the background. A particularly difficult problem is how to track a target which moves through scenes featuring several other very similar objects, or which moves past extremely cluttered or camouflaged image regions, shown in figure 1.3. In these situations, it is difficult to distinguish the target using the appearance information solely obtained from the target bounding box. Additional information, such as dynamic models of the target and nearby distracting image regions in the scene, may be useful to support robust tracking.

1.2 Objectives

To tackle the above mentioned problems, we aim at: 1) designing a robust tracker in RGB images which could adaptively fuse and update multiple features to overcome a variety of difficult tracking problems, e.g significant target deformation, rapid and erratic target motion, illumination change and viewpoint change; 2) extending the proposed RGB tracker to handle RGB-D

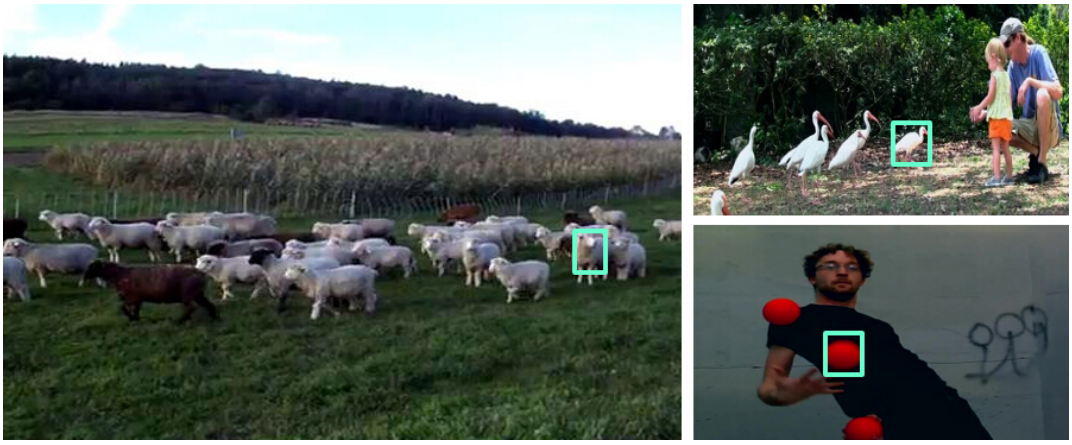

Figure 1.3: Sequences with many distractor objects, used in our experiments. Bounding box is the single target we want to track. It is extremely difficult to distinguish one true target out of many identical contextual regions.

images by showing how depth information can be robustly combined with RGB information during tracking; 3) exploiting the contextual information which could handle camouflage problems in the highly cluttered scenes and providing a deeper analysis of the tracking performance under those severe situations.

1.3 Contributions

In the light of the objectives in section 1.2, we propose three novel methods which address these problems: 1) we build multi-layer appearance models, and combine them with clustered decision trees to tackle the single target tracking problem in RGB images; 2) we extend this RGB tracking approach to also handle RGB-D data, by exploiting the depth feature with embedded common-sense knowledge to handle highly dynamic depth feature variation; 3) we show how to exploit contextual information, by detecting, learning and tracking distractors, to support single target tracking in extremely cluttered scenes.

We first build a single target tracking algorithm, which achieves state-of-the-art robustness by addressing the problems in three fundamental areas: target representation, matching methodology, and model adaptation. The target is *represented* at three different layers: pixel layer, local-parts layer and target layer, shown in figure 1.4. A novel "Adaptive Clustered Decision

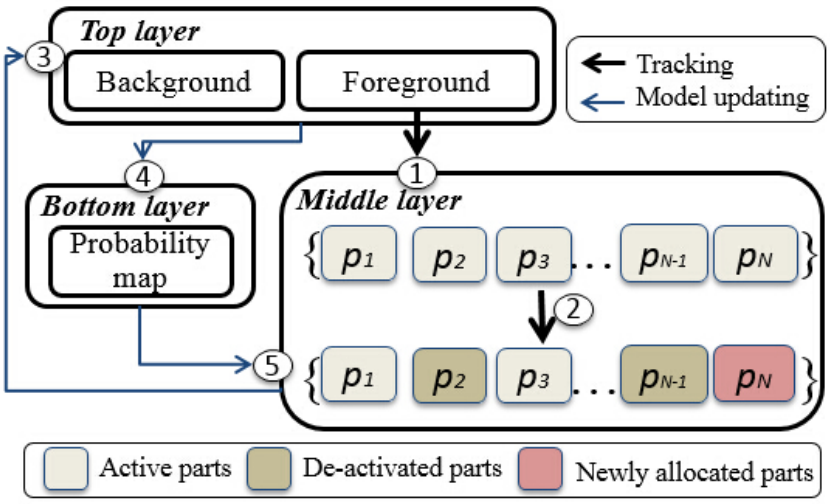

Figure 1.4: Block diagram of the proposed tracking process. 1- tracker propagation by the top layer foreground information matching; 2- middle layer (target part) matching with clustered decision tree; 3- update the top layer model; 4- feed back the bottom layer (pixel) feature; 5- re-sample drifting parts at the middle layer model [1].

Tree" method is proposed to enable robust *matching* between successive frames by dynamically selecting the minimum combination of necessary features, thereby providing robustness with computational efficiency. The target model is *updated* through a closed loop with cross-constraints between different model layers. We test the tracker using two different tracking benchmarks [2, 3], based on two different test methodologies, and show it to be more robust than all other state-of-the-art methods from those tracking challenges while also demonstrating competitive tracking precision [1].

We extend the proposed 2D RGB tracker to 3D RGB-D images exploring a more robust way of applying the depth information. Note that the depth information encodes the latent physical constraints of the object layout which can provide a unique feature for RGB-D tracking, which is unavailable for RGB tracking. Therefore, a physical constraint of the target with respect to other objects in the depth context is exploited, to adaptively estimate the range interval occupied by the target. This physical constraint encodes common-sense knowledge that the target parts cannot move behind background regions or in front of occluder. Despite the availability of several high quality benchmark datasets for RGB tracker evaluation [3, 21, 22, 23], those available for RGB-D tracker evaluation are far more limited. Therefore, we also construct a new RGB-D

dataset with extensive bias analysis to facilitate the evaluation in RGB-D images.

We exploit contextual information to solve the problem of tracking a single target which is camouflaged against scenes containing a large number of similar non-target entities, which we call *distractors*. We first propose a multi-level clustering method for online detection and learning of multiple distractors. To distinguish the target from these distractors, we use global dynamic constraints (derived from target and distractor information) to optimize the estimated target trajectory. To gain a deeper understanding of tracker performance on cluttered scenes, we propose a new set of sub-attributes to describe different kinds of cluttered scenes, and augment publicly available benchmark data by per-frame annotating all frames of all sequences with all sub-attributes. Using this dataset, we show that our proposed method outperforms the best ranked state-of-the-art single target trackers on highly cluttered scenes. We also show that the proposed method outperforms state-of-the-art multi-target trackers, in scenes where the true target moves among many other similar objects, even though the multi-target methods are given significant additional a-priori information about distractors.

For clarity, we itemise the contributions in terms of the tracking algorithms and datasets below:

A. Tracking algorithms

- An RGB tracker using adaptive clustered decision trees and dynamic multi-layer appearance models (chapter 3).
- An RGB-D tracker exploiting multi-feature local-global spatio-temporal consistency (chapter 4).
- Distractor-supported single target tracking under extremely cluttered scenes (chapter 5).

B. Tracking datasets

- A new RGB-D benchmark dataset with extensive bias analysis (chapter 4).
- An augmented RGB tracking benchmark dataset with proposed sub-attributes for the cluttered scenes (chapter 5).

1.4 Publications arising from this work

A. Peer-reviewed full-length papers(first-author)

- **J. Xiao**, R. Stolkin, M. Oussalah, A. Leonardis. Context based model adaptation for tracking with online weighted color and shape feature. IEEE sensors journal, 2016. (related to chapter 3)

- **J. Xiao**, M. Oussalah. Performance Evaluation of Particle Filter Based Visual Tracking. Journal of Intelligent and Fuzzy Systems, 2016. (related to chapter 3)

- **J. Xiao**, R. Stolkin, A. Leonardis. Single target tracking using adaptive clustered decision trees and dynamic multi-level appearance models. CVPR , Boston, U.S.A, 2015. (related to chapter 3)

- **J. Xiao**, M. Oussalah. Collaborative Tracking for Multiple Objects in the Presence of Inter-occlusions. IEEE Transactions on Circuits and Systems for Video Technology, 2015. (related to chapter 5)

- **J. Xiao**, R. Stolkin, A. Leonardis. Multi-target tracking in team-sports videos via multi-level context-conditioned latent behaviour models. BMVC , U.K., 2014. (related to chapter 5)

- **J. Xiao**, M. Oussalah, Robust model adaptation for tracking with online weighted color and shape feature, 4th IEEE International Conference on Image Processing Theory, Tools and Applications, France, 2014. ***Best poster award in BMVA summer school.*** (related to chapter 3)

- **J. Xiao**, R. Stolkin and A. Leonardis, An enhanced adaptive coupled-layer LGTracker ++, ICCV Visual Object Tracking Workshop. 1-8 December, Sydney, Australia, 2013. ***IEEE PAMI TC Conference Travel Award.*** (related to chapter 3)

- **J. Xiao**, M. Oussalah, Scale Modification through Particle Distribution in Colour Based Tracking, The 10th European Conference on Visual Media Production. 6-7, November, London, UK, 2013. (related to chapter 3)

B. Peer-reviewed full-length papers(co-author)

- N. Marturi, V. Ortenzi, **J. Xiao**, M. Adjigble, R. Stolkin and A. Leonardis, A real-time tracking and optimised gaze control for a redundant humanoid robot head, IEEE-RAS International Conference on Humanoid Robots, 2015. (related to chapter 3)

- H. Haggag, M. Hossny, S. Haggag, **J. Xiao**, S. Nahavandi, D. Creighton, LGT/VOT tracking performance evaluation of depth images. 9th International Conference on System of Systems Engineering, 2014. (related to chapter 4)

- M. Kristan et al, The visual object tracking VOT2013 challenge results, ICCV workshop, 2013. (related to chapter 3)

1.5 Overview of the dissertation

The organisation of the thesis is as follows:

Chapter 2 presents a literature survey, covering RGB tracking algorithms, RGB-D tracking algorithms, tracking benchmark datasets and dataset bias analysis. This chapter reviews RGB trackers first in terms of target representation, matching methodology, and model adaptation. For RGB-D trackers, we mainly analyse the different ways of utilizing depth information in the literature. The representative tracking benchmark datasets are also summarized here, while bias analysis is provided since it can significantly effect conclusions drawn from performance evaluation of tracking methods.

Chapter 3 presents a new single target tracker with multi-layer appearance model. An adaptive clustered decision tree method is proposed to select the minimum combination of features necessary to sufficiently represent each target part, thereby providing robustness with computational efficiency. Based on two different evaluation protocols, we have tested the tracker using two different tracking benchmarks [2, 3] with total 70 sequences and comparing against more than 50 other trackers from the literature. The proposed tracker is demonstrated to be significantly more robust than the state-of-the-art methods, while also offering competitive tracking

precision [1].

Chapter 4 focuses on single target tracking in RGB-D images, where an adaptive multi-feature local-global target model with both temporal and spatial constraints is proposed. In the bounding box layer, temporal consistency is used to adaptively fuse information from both RGB and depth images to find a candidate target region. In frames where one or more features are temporally inconsistent, this global candidate region is further split into local candidate regions for matching to target parts. We derive a physical constraint from the depth context, which robustly accommodates large target depth variation, while still enabling accurate reasoning about occlusions. For evaluation, we introduce a new RGB-D benchmark dataset with per-frame annotated attributes and extensive bias analysis. Our tracker is evaluated using two different methodologies [2, 3] and in both cases it outperforms other state-of-the-art RGB-D trackers.

Chapter 5 shows how distracting contextual regions can be detected, learned and explicitly exploited to support the single target tracker under conditions of extreme clutter and camouflage, including frequent occlusions by objects with similar appearance to the target. A multi-level clustering method is proposed for online detection and learning of multiple target-like regions, called distractors, when they appear near to the true target. To distinguish the target from these distractors, we use global dynamic constraints (derived from target and distractor information) to optimize the estimated target trajectory. To evaluate our tracker, we have augmented publicly available benchmark videos, by proposing a new set of clutter sub-attributes, and annotating these sub-attributes for all frames in all sequences. Using this dataset, we show that our proposed method outperforms other state-of-the-art single target trackers on highly cluttered scenes. We also show that our proposed method outperforms state-of-the-art multi-target trackers, in scenes where the true target moves among several other similar objects.

Chapter 6 summarizes the contributions of this thesis, points out the problems of existing methods and discusses the potential avenues for future work.

CHAPTER 2

LITERATURE REVIEW

Single target tracking, as an important component in the tasks of motion-based recognition, automotive surveillance and vehicle navigation, has been extensively studied in recent decades [24]. Even though great effort has been paid in this area, visual object tracking is still a challenging task, because of background clutter, occlusion, fast movement, illumination changes, object scale change and deformation. Therefore, this chapter provides a survey of the research that has already been published in relation to the single target tracking in RGB images (section 2.1), RGB-D images (section 2.2), and highly cluttered scenes where need proper context understanding (section 2.3). The state-of-the-art tracking benchmark datasets for algorithm evaluation are also reviewed (section 2.4).

2.1 RGB trackers

Based on the prior information about the target category, the tracking algorithms are usually classified as category-free methods, like KCF [9], Struck [25], LGT [6], and category-based methods, like pedestrian tracking [26], vehicle tracking [27], hand tracking [28]. Category-free tracking methods are usually initialized using a single bounding box/ellipse in the first frame without any other pre-trained information, while the category-based tracking methods usually require a training process for one particular object category. As the category-free tracking methods are acknowledged for the simple initialization and easy generalization among different

object categories, our work also focuses on tackling the problems in this particular tracking research branch - category-free tracking methods.

Although significant progress has been made in this area [2, 3], the most sophisticated trackers are still prone to failure in challenging scenarios, including environmental clutter and camouflage in one or more feature modalities, rapid and erratic target motion, occlusions, significant appearance changes and shape deformation over time. In general, a robust tracking system consists the following three parts [1]: 1) the appearance model of the target; 2) the mechanism for matching model parts to image regions at each frame; 3) the mechanism for continuously relearning or updating models of targets which change their appearance over time. Therefore, we review the related works with respect to the target representation, matching methodology and model updating method accordingly.

A. Target representation

A crucial component of a tracker is the target model that represents the object status during tracking, i.e. position, scale, and speed etc. Based on the different representations, tracking methods are usually categorized as: *point* tracking, *kernel* tracking, and *silhouette* tracking [29], shown in figure 2.1.

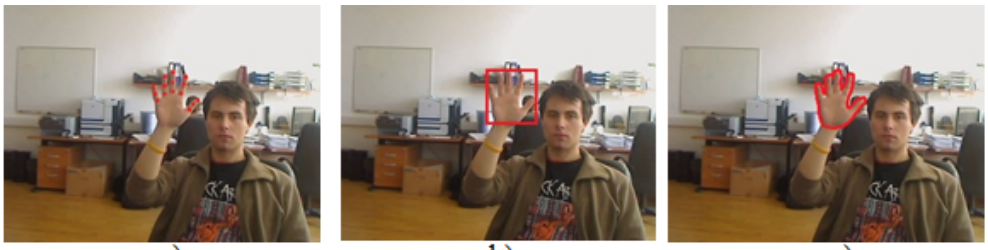

Figure 2.1: Different representations of a target: a) point tracking; b) kernel tracking; c) silhouette tracking.

For *point* tracking [30, 31, 32, 33], a moving object is represented by some feature points detected across consecutive frames during the tracking. The problem here is formulated as matching the point correspondence between each frame. Usually, the point tracking methods are quite sensitive to noise and are easily distracted under conditions of background clutter. To achieve better performance, kinematic constraints are often imposed in some particular applications [34]. Although this enables good performance in particular conditions, such trackers have

difficulties to be generalised to other tracking conditions, e.g. the kinematics built for a human cannot be used for a snake.

Silhouette [35, 36, 37, 38, 39] tracking provides the accurate shape description (contour) for the target in each frame. Usually, those methods iteratively evolve the initial contour by minimizing the contour energy using gradient descent, which is applied to track the deformable targets including animals and human hands etc.

Kernel [40, 41, 42, 43, 44] is a region-based representation, which can be a rectangular or an elliptical template with an associated histogram, usually referring to the target shape and appearance. Kernel based tracking is conducted by computing the motion of the kernel that explicitly defines the target region in the next frame.

In this thesis, we utilize a bounding box to represent the target and focus on solving the prior information free tracking problems. Therefore, we mainly review the related works of kernel based tracking methods. Based on the image data inside the target kernel, the algorithm converts the raw data into some more informative features for tracking [15], which can include:

- **Colour features.** The colour of an object is a widely used descriptor for target representation, which can be defined differently according to the various colour spaces, namely RGB, Log-RGB, Lab, HSV and HSL [45]. As the colour is influenced by spectral power distribution of the illumination and the surface of reflectance properties of the object, these colour spaces are sensitive to stray light and are also vulnerable to the changes of the lighting conditions [29].

- **Haar-like features.** A Haar-like feature [46, 47], i.e. edge feature, line feature and center-surrounded feature, encodes the existence of oriented contrasts between adjacent rectangular regions at a specific location in a detection window. It sums up the pixel intensities inside each region, where these sums are used to calculate the difference between regions. Haar-like features are acknowledged for their good trade-off between speed and accuracy [48].

- **Binary pattern features.** Local Binary Pattern (LBP) is a simple but very efficient texture

operator which utilizes a non-parametric kernel (3*3 kernel) to summarize the spatial structure of an image [49, 50]. LBP operator utilizes the greyscale of the central pixel to label its neighbourhood pixels with 0 or 1, where that central pixel is represented by a binary string from its neighbours, which provides both computational simplicity and discriminative power (robustness) to monotonic gray-scale changes caused, for example, by illumination changes [51, 52].

- **HOG features.** The histogram of gradients (HOG) feature, proposed by [53], uses a distribution of gradient intensities in terms of edge directions to represent object shape. The candidate region is divided into small connected cells (regions) with the following steps: gradient computation, orientation binning, and descriptor blocks. Since HOG feature operates on the local cells, it is invariant to geometric and photometric transformations.

- **SIFT features.** Scale Invariant Feature Transform (SIFT) feature is composed of the following four steps: 1) convolve the image with Gaussian filters at different scales, where the candidate interest points are selected from the minima and maxima of the Difference-of-Gaussians (DoG) images across scales; 2) update the candidate location by interpolating the color information of neighbourhood pixels; 3) eliminate low contrast candidate points; 4) assign orientations to the remaining interest points due to the peaks in the histograms of gradients.

- **Bag of words features.** Bage of words (BoW) model is a sparse vector of occurrence counts of independent image features (words) [54]. Extracting the BoW features from images usually includes the following steps: 1) detect key point based features, i.e. SIFT features; 2) compute local descriptors over those key point based features; 3) quantize the descriptors into words to generate the visual vocabulary, and 4) construct a histogram of word occurrences in the image (codebook), which contains the presence or absence information of each visual word.

- **Sparse features.** The goal of sparse coding is to approximate input signals by a linear combination of weighted bases(atoms), which is applied to learning over-complete dictionary

of atoms which captures high-level patterns in the input data [55, 56]. However, to learn the over-complete dictionary and solve the corresponding optimization problems is still a significant challenge with respect to the computational cost [57, 58].

- **CNNs features.** A Convolutional Neural Networks (CNNs) is comprised of small neuron collections across multiple layers [59, 60], where those neuron collections are tiled in each layer with the shared weights to better represent the original image. As the same filter is used for each pixel in the layer, the CNNs features can tolerate image translation. Some ways to train CNNs include filter shape, the number of filters, and max pooling shape.

The above features are extracted from the particular kernel for matching during tracking. In terms of the kernel scale, the target representation methods are categorized into three groups: *holistic* [61, 62, 63, 64, 65, 66], *local* [67, 68, 40, 69, 70], and *hybrid* [6, 71, 72, 7, 73, 74, 75]. A holistic representation reflects the overall statistical characteristics of the target in a simple and computationally efficient model. However, the holistic template based trackers have difficulty in handling significant appearance changes and deformations of the target. In contrast, a local feature representation provides more flexibility for deforming target matching but usually with increased computational cost. Recent works use both holistic and local templates to represent the target in a hybrid way which compensates the disadvantages of holistic or local representations.

B. Matching methodology

To estimate the state of the target, the algorithm must match observations from a candidate image region to the target representation model [1]. Here, we denote the candidate feature vector as q_{can} while the reference feature vector as q_{ref}. The common methods of comparing the candidate feature to the reference feature are reviewed below:

- **Bhattacharyya distance.** The Bhattacharyya distance measures the dissimilarity of two probability distributions (feature vector or histogram) bin-by-bin [76, 77]. The distance $D_B(q_{can}, q_{ref})$ is computed by: $D_B(q_{can}, q_{ref}) = \sqrt{1 - \sum_{\xi=1}^{m} \sqrt{q_{can}(\xi) q_{ref}(\xi)}}$, where

m is the number of the bins inside the corresponding feature vector.

- **Euclidean distance.** The Euclidean distance $D_E(q_{can}, q_{ref})$ between candidate feature q_{can} and the reference feature q_{ref} is the length of the line segment connecting them [78]. A generalized term for the Euclidean norm is the L_2 norm or L_2 distance, computed by $D_E(q_{can}, q_{ref}) = \sqrt{\sum_{\xi=1}^{m}(q_{can}(\xi) - q_{ref}(\xi))^2}$.

- **Earth movers distance.** Earth Movers distance (EMD) was defined by Rubner et al. [79] as the minimal cost that must be paid to transform one feature vector into the other feature vector, which is cross-bin similarity measurement computed by: $D_{EMD}(q_{can}, q_{ref}) = \frac{\sum_{\xi_1=1}^{m}\sum_{\xi_2=1}^{m} d(\xi_1,\xi_2)f(\xi_1,\xi_2)}{\sum_{\xi_1=1}^{m}\sum_{\xi_2=1}^{m} f(\xi_1,\xi_2)}$, where $f(\xi_1, \xi_2)$ denotes the flows. Each $f(\xi_1, \xi_2)$ represents the amount transported from bin ξ_1 supply to the bin ξ_2 demand, $f(\xi_1, \xi_2) = |q_{can}(\xi_1) - q_{ref}(\xi_2)|$. $d(\xi_1, \xi_2)$ is called the ground distance between different bins, $d(\xi_1, \xi_2) = |\xi_1 - \xi_2|$. All above, ξ_1, ξ_2 ranges from $[1, m]$, while m is the number of bins.

- **Correlation filters.** Correlation filters measure the similarity between the image template and the image candidate region to produce the correlation peaks for the target regions while low response for the background [80]. Correlation filters take advantage of the fact that the correlation operations of two image regions could be conducted by element-wise multiplications using Discrete Fourier Transform, and have proved to be competitively efficient at hundreds of frames-per-second [81]. Correlation output is transformed back into the spatial domain as a spatial confidence map using the inverse Fast Fourier Transform, where the region with the maximum value indicates the position of the target.

- **Logistic regression.** Logistic regression (LR) [82] is a statistical method to analyse a (training) dataset and find the best fitting model with independent variables that determine an outcome (matching probability). During tracking, given the features of a candidate region, LR computes the region's probability of being a target according to the logit transformation (trained fitting model).

- **Adaptive Boosting.** Adaptive Boosting [83] involves starting with a single weak classi-

fier, and recursively adding the opinions of additional weak classifiers, with confidence weights for each successive classifier being chosen to minimise the overall error with respect to ground-truthed training data. [84, 85] use individual bins of a feature vector as "weak classifier" and employ Adaboost as a way of creating a weighted combination of these classifiers which best discriminates between target and surrounding background regions at each frame.

- **Decision tree classifiers.** Probabilistic Boosting Trees [86] have been used for classification, recognition, tracking and clustering problems, which are binary discriminative classifiers. Each node of the tree combines weaker classifiers (lower nodes), yielding a top node which outputs an overall posterior probability (of matching) by integrating the conditional probabilities gathered from its sub-trees.

- **Support vector machine.** Mapping the original finite dimensional space into a higher dimensional space makes the data classification easier. Support Vector Machines (SVM) try to find a hyperplane that has the largest distance (functional margin) to the nearest training-data point. The vectors that define the hyperplanes are linear combinations with parameters of images feature vectors [87]. Given the training data, an SVM algorithm builds the model to determine whether the candidate region contains the target.

In the early tracking literature, tracking was often conducted using only a single feature [64, 64, 88, 66], which was not sufficient to handle large appearance variations. More recent works [89, 90, 91, 74] increasingly exploited combinations of multiple features. Based on the method of feature fusion, matching methods can be divided into two categories: *generative* methods [92, 93, 94, 95, 96] and *discriminative* methods [97, 98, 61, 99, 100, 90, 91]. The generative methods treat the features in a homogeneous way. Simplistic approaches of merely multiplying the contributions of each feature modality, i.e. as a product of probabilities (or a product of experts [101]) will be prone to failure in scenes where one or more of the feature modalities is challenged by clutter of similar appearance to the target. This is for the simple reason that multiplication of feature weights can cause a false negative detection by a poorly

performing feature. In contrast, the discriminative methods aim at distinguishing highly informative features and maximising the separability between the object and background with a weighted combination of multiple features, which often improves the tracking efficiency and robustness.

C. Model adaptation

For robust tracking, it is essential to continuously update or relearn the target model to cope with appearance changes. An appropriate target model should enable the tracker to overcome errors in the relearning process which might corrupt the target model, and support long term tracking without drifting [1, 63]. In general, the model adaptation methods can be categorized as *autoregressive adaptation* and *template replacement*.

- **Autoregressive adaptation.** Works such as [64, 67, 7, 71] updated the model as a simple linear combination of the previous model and the most recent estimation of the target region in the current image.

- **Template replacement.** For those classifier based tracking algorithms [40, 25, 73], the positive or negative samples used in the training stage are replaced by the new observations, where those observations are often chosen in terms of matching probability or discriminability.

In early work [64], the model adaptation was solely based on the target observation. Without additional methods for precise delineation of the target parts or background, if a few background pixels are erroneously learned into the target model, then tracking will deteriorate, resulting in even more erroneously relearned pixels, and exponentially increasing tracking instability [1]. In MIL [102] and other trackers such as OB [103] and SB [84] updating of the target model is performed by an evolving boosting classifier that tracks image patches and learns the object appearance. Interestingly, OB [103] can be regarded as a non-Bayesian approximation to the simpler Bayesian ABC method [66] but which enables continuous relearning of the target model. However, online boosting requires that the data be independent and identically distributed, which is a condition not satisfied in most real video sequences, where data is often

temporally correlated [89]. A more robust updating mechanism is achieved by [6], which forms a cross constraint paradigm to stably constrain the relearning of a two-layer target model - global (bounding region) and local (parts based) models are used to constrain (and thereby stabilise) each others' online relearning. However, this method (and most earlier methods e.g. [64]) updates target appearance models at a fixed rate, regardless of the confidence (or lack of) in current target observations. We summarize the most recent published representative tracking work in table 2.1.

Based on the literature study, we have proposed a novel method for RGB target tracking in chapter 3, comprising a multi-layer target model and an adaptive clustered decision tree method for both matching and relearning middle layer target parts at successive image frames.

2.2 RGB-D trackers

To further improve tracking performance, recent works have tried to incorporate additional features [16, 73] , with depth information attracting growing interest [5]. While RGB features can be comparatively invariant to target motion, depth information can vary rapidly. To exploit the depth information in a robust way, prior information about the target model is employed for category-based tracking [113, 114, 115, 116, 117]. However, such methods still have difficulty in robustly modelling the depth information for deformable targets, i.e. deformable objects, which limits its popularity for the category-free tracking research.

Chen et al. [19] proposed an RGB-D object tracking algorithm by applying kernel partial least squares analysis to learn the discriminative appearance model of the target. The depth information was used to detect occlusion during tracking. They first assumed that the object depth dominated the bounding box. Then, the occlusion was confirmed when the mean depth of the detected bounding box exhibits a sudden change. Intuitively, the tracker fails when the target scale is not precisely estimated or the target rapidly moves towards or away from the camera.

Cao et al. [18] proposed an RGB-D tracker with a patch-based appearance model. They

Table 2.1: Functional comparison of tracking algorithms. **Features**: IH: intensity histogram; OP: optical flow; CE: convex envelope; BP: binary pattern; HOG: Histogram of oriented gradients; SR: sparse representation; BRISK: Binary robust invariant scalable keypoints; CNN: convolutional neural networks. **Matching**: BC: bhattacharyya coefficient; BI: bayesian inference; EMD: earth movers distance; CF: correlation filter; NNC: nearest neighbour classifier; SSVM: structure output SVM. **Model adaptation**:TR: Template replacement; AU: autoregressive update; IU: incremental update; EM: error minimization.

Template	Tracker	Features	Matching	Model adaptation
Local	Frag [69]	IH	EMD	N/A
	BHMC [67]	IH	BI	AU
	SOWP [40]	IH,HOG	SSVM	TR (budget based)
	DLR [98]	SR	L_0 regularization	TR (time based)
	RANSAC [104]	BRISK, OP	ED	AU
	OLF [89]	SIFT	NNC	TR (ranking based)
	RPT [105]	HOG, motion	BI	TR (distribution, computation, likelihood)
	PACF [106]	HOG	CF	AU
Holistic	KCF [9]	HOG	CF	AU
	MKCF [107]	HOG,IH	CF	AU
	LKCF [108]	HOG,IH	CF	AU
	CPF [88]	IH	BC	AU
	IVT [109]	covariance matrix	BI	IU
	Struck [25]	Haar	SSVM	TR (budget based)
	TLD [110]	BP	NNC	P-N learning
	DCMT [111]	IH	BI	AU
	L1 [112]	SR	L_1 regularization	TR (ranking based)
Hybrid	LGT [6]	IH, OP, CE	BI	Lobal-glocal interaction
	SCM [7]	SR	L_1 regularization	TR (ranking based),AU
	FCNT [72]	CNN	L_1 regularization	EM
	HCNT [71]	CNN	CF	AU
	SST [73]	SR	$L_{2,1}$ regularization	TR (ranking based)

designed a robust estimator to fuse the votes from the local patches (for target location) by filtering the most similar and dissimilar candidate patches. As their method simply partitioned the holistic template by fix grid to form the local patches, it has difficulty handling the object with significant deformation. Additionally, their method relied on RGB data for target estimation, while using depth information solely for reasoning about occlusions. The target can become mislabelled as an occluding object if it moves rapidly towards the camera.

Sophisticated RGB-D work was carried out by Song et al. [20]. They proposed several different RGB-D trackers, using both 2D and 3D models, to evaluate the extent to which the addition of depth information could improve overall tracking performance. In their work, the best tracking performance was achieved by using an RGB-D HOG feature detector and a point cloud feature detector. The detection results of each feature are summed and adjusted with 3D iterative closest point tracking for the final tracker estimation. [20] propose that occlusions be detected when the target region depth histogram develops a newly rising peak with a smaller depth value than the target. Similar to the problems of [19, 18], this heuristic only holds true when the target estimation is highly accurate and the depth information is relatively stable.

Due to the success of Kernelized Correlation Filters (KCF [9]) in RGB tracking with respect to high accuracy and processing speed, Camplani et al. [10] applied kernel correlation filters to RGB and depth maps respectively. They modelled the target depth distribution with a Gaussian distribution, where changes in depth were utilised to adapt the scale and infer occlusion. This holistic template based tracker has difficulty handling significant target deformations. Moreover, relying solely on depth data for occlusion reasoning can cause failures.

Meshgia et al. [11] proposed an "occlusion aware" particle filter in a probabilistic framework to handle complex and persistent occlusions in RGB-D images, which employs a latent variable representing an occlusion flag for each particle. Seven features were fused to measure the similarity between the candidate and the target model. However, it uses a pre-defined depth threshold which has difficulty tracking targets which exhibit significant depth variation. Also, they fused all the features in a homogeneous way, where one poorly performing feature can damage the overall tracking performance.

The common problem of mentioned works such as [19, 18, 10, 11, 20] is that they rely on the depth information solely for occlusion reasoning in a simple way, either by measuring depth changes in mean value or in Gaussian distribution of the target depth. Intuitively, rapid motion of the target towards or away from the camera can easily lead to failure. Therefore, it is essential to exploit depth feature in a robust way.

Note that rather than just treating depth information as an *image* feature, it also reflects the *physical* constraint (layout) in the spatial world. Instead of the heuristic assumptions about stability of depth information, the method proposed in chapter 4 encodes common-sense knowledge that target parts cannot be located behind background regions or in front of occluding objects, which robustly accommodates large target depth variation while still enabling accurate reasoning about occlusions.

2.3 Tracking by exploiting scene distractors

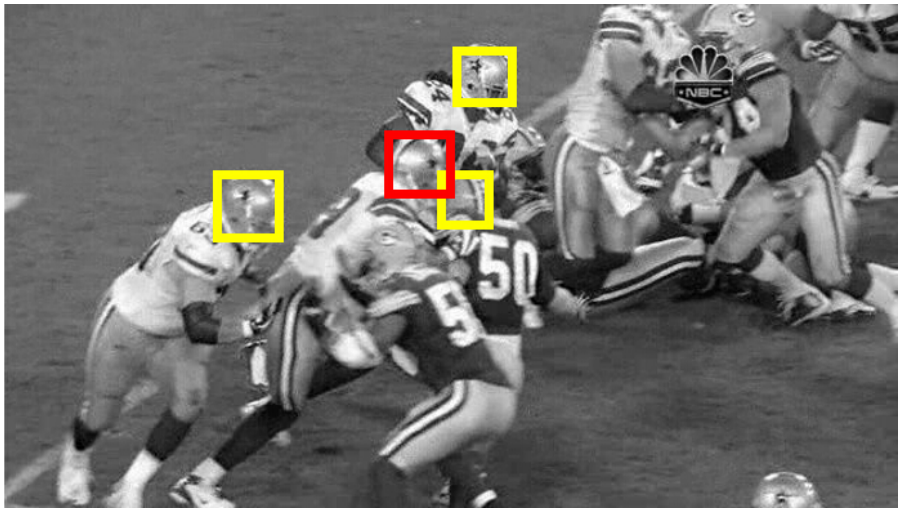

Figure 2.2: Target moves past extremely cluttered or camouflaged image regions, called "distractors". Target and the distractors have identical appearance, where it is hard to distinguish the target solely based on the appearance. Red bounding box: the real target; yellow bounding boxes: distractors.

Based on the initialized bounding box and later target observation, a well designed tracker can overcome some conventional challenges, like significant target deformation, illumination change and fast motion. A particularly difficult problem is how to track a target which moves

through scenes featuring several other very similar objects, or which moves past extremely cluttered or camouflaged image regions, shown in figure 2.2. In these situations, it is difficult to distinguish the target using only target appearance information [13], since the background regions of the image may share similar appearance information. To achieve successful tracking, the tracking algorithm needs to go beyond the target information and "interact" with the context in a sophisticated way, understanding where is the potential distraction in the scene, which we call "distractors". Note that detecting distracting image regions before they come close and distract the tracker can actually support robust tracking by modelling the dynamic behaviour of such regions. Therefore, in this thesis, we try to exploit additional contextual information to solve the problem of tracking a single target which is camouflaged against scenes containing a large number of similar non-target entities.

In scenes with clutter and camouflage, the method proposed in this thesis detects multiple target-like regions and explicitly associates one region to the target, which may at first sight appear analogous to the data association approach in multi-target tracking. However, note that our proposed method is *not* a multi-target tracker (figure 2.3): i) it only identifies the location of a single target at each frame; ii) it does not assign individual IDs to multiple other objects; iii) it is initialised with only a single target bounding box, and must automatically detect and learn any other target-like objects (distractors) on-the-fly.

We first review work on *single*-target tracking in highly cluttered scenes, which explicitly exploit the contextual information. We also review some *multi*-target tracking literature, since our proposed method (global dynamic constraints) for handling distractors, is somewhat related to the data association methods used in multi-target tracking.

- **Context based single-target tracker.** To track a single target robustly against clutter, the tracker should learn and exploit contextual information. [118, 119] observed that non-target objects, known as *supporters*, may sometimes be associated with a target and can be used to help infer its position. However, in highly cluttered scenes, it may not be possible to find supporters that persistently move around the target with strong motion correlation. [120, 121, 122] detected distractor regions with similar appearance to the tar-

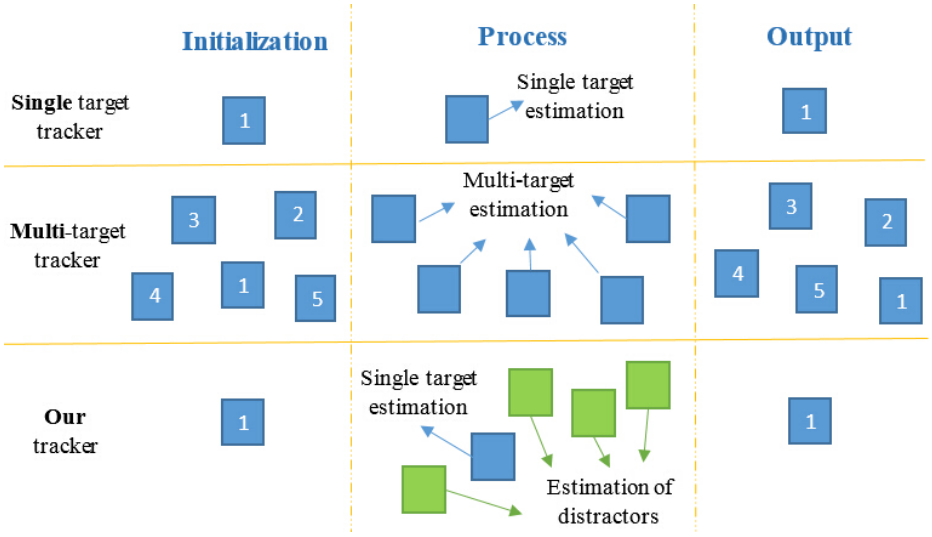

Figure 2.3: Relationship between single target trackers, multi-target trackers and our proposed tracker. Single target trackers do not explicitly model information about distractor objects. Multi-target trackers model and identify multiple target regions, exploiting additional initialized prior knowledge. In contrast, our tracker is initialised the same as single-target trackers, while detecting and tracking other distractor objects on-the-fly.

get. However, their tracking accuracy heavily depends on the pre-defined spatial density of samples, where sparse sampling cannot adequately distinguish adjacent objects, and dense sampling is computationally expensive. Even with dense sampling, such methods can still fail to distinguish adjacent or overlapping objects. Note that such methods [120, 121, 122] maintain multiple image regions as target candidates, but do not specifically decide which region is the target at each frame. [8] tried to distinguish the target from distractors by using an appearance matching score. However, such algorithms are prone to failures when the target is camouflaged, occluded or undergoing deformation, when appearance matching methods can cause the tracker to erroneously fixate on clutter. [111] recently proposed a distractor-aware target model to select salient colours in single target tracking. In contrast, our work addresses video sequences that are so extreme that target and distractors may have *identical* appearance (illustrated in figure 2.2), and cannot be disambiguated by any appearance features. Moreover, [111] distinguishes the target from distractors based on spatial distance between the current observations (k frame) and the previous target estimate ($k-1$ frame). This spatial distance method can fail when the camera itself undergoes significant motion. In contrast, we propose a global

dynamics model for the target and nearby distractors, and show that this enables robust tracking even in sequences filmed from moving cameras.

- **Data association in multi-target tracker.** Our proposed global dynamics model is related to, but distinct from, the problem of data association discussed in the multi-target tracking literature. Berclaz et al. [123] reformulated the data association problem as a constrained flow optimization convex problem, solved using a k-shortest paths algorithm. However, the computational cost of generating k paths is quite high, especially for our problem of finding a single target at each frame. Moreover, this method first obtains detections for every frame throughout an entire video sequence, and then mutually optimises the target IDs over all frames. Such post-processing methods cannot be used for online target tracking. Shitrit et al. [124] also relaxed the data association problem as a convex optimization problem which explicitly exploited image appearance cues to prevent erroneous identity switch. However, appearance cues are not sufficiently discriminating to distinguish between the target and distractors in the extremely challenging videos, i.e. a target moves through scenes featuring several other very similar objects, or which moves past extremely cluttered or camouflaged image regions, which we aim to tackle in this work. [13] utilized motion dynamics to distinguish targets with similar appearance, in order to reduce instances of target mislabelling and recover missing (occluded) data. However, their algorithm requires the number of targets to be known a-priori. In contrast, our method handles the situation where the number of distractors is unknown and has to be learned on the fly during tracking. [125] proposed a constrained sequential labelling to solve the multi-target data association problem, which utilized learned cost functions and constraint propagation from captured complex dependencies. However, their approach is only designed to handle the case of piece-wise linear motion. [12] developed an online multi-target tracker which extended the single target tracker of [25] by using global data association. However, their proposed candidate regions must be densely sampled which can be extremely computationally expensive. Moreover, their global identity-aware network flow graph depends heavily on target appearance models, which have difficulty in

handling highly cluttered scenes, especially where both target and distractors share identical appearance. Our early work [126] learned and exploited the global movement of sports players to inform strong motion priors for key individual players. Information from the global team-level context dynamics enabled the tracker to overcome severe situations such as inter-player occlusions. However, this context-conditioned latent behavior model do not readily generalise to non-sports tracking situations.

In this work, we further encode the contextual information in chapter 5 to support the single-tracker that goes beyond the target itself, which perceives distractors and models (understands) their dynamics in the scene, shown in figure 2.4.

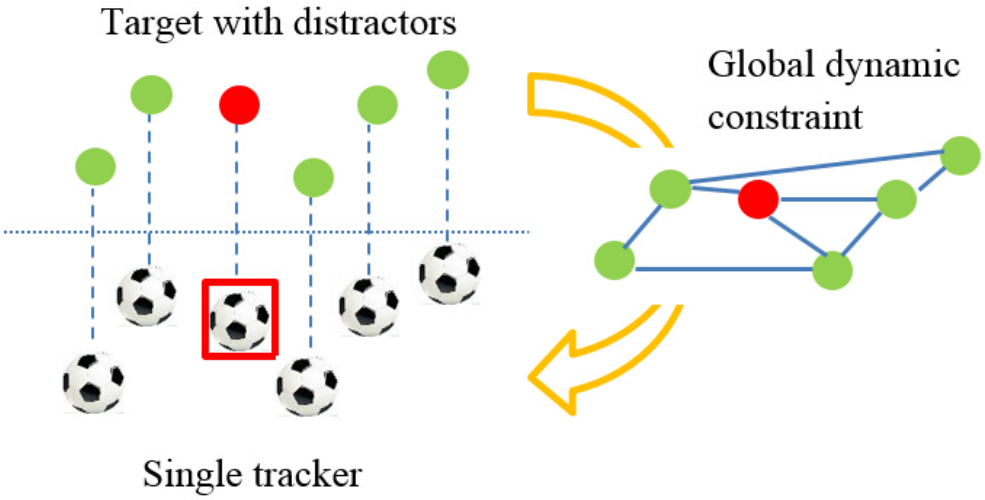

Figure 2.4: The proposed framework using contextual information. The red bounding box/dot represents the target while the green colour indicates the distractors. The algorithm simultaneously track the target and distractors, where the information is built in the global dynamic constraint to further improve the single target tracker.

2.4 Dataset construction and annotation

Benchmark datasets have played a leading role in advancing tracking research by facilitating the evaluation process. For persuasive comparisons of visual tracking algorithms, constructing test data should consider two questions: which situations should be included and how should the algorithms be tested to reach a convincing conclusion [127]. After several decades of efforts,

several benchmark datasets have been created for category-free trackers. Here, we first review the related work with respect to the dataset construction and attributes annotation. Since the bias of the dataset significantly affects the algorithm evaluation, we also analyse the existing methods for bias analysis.

2.4.1. Dataset construction and attributes annotation.

To build a test dataset for category-free single-target trackers, it is essential to include scenes broadly and comprehensively that represent the world. One representative work is that of Smeulders et al. [128] who constructed an RGB tracking dataset ALOV++ with 314 sequences and evaluated the algorithms by survival curves, Kaplan Meier statistics, and Grubs testing, with the evaluation metrics F-score and object tracking accuracy (OTA) score. Since ALOV++ has no attributes annotations, it cannot explain tracker strength or weakness for particular functional conditions, such as occlusion, illumination change etc. A landmark RGB benchmark, VTB, initially contained 50 videos [23], later extended to 100 videos [129] with per-sequence binary attributes labels according to the intuition of human annotators. They proposed to test the algorithms in terms of temporal robustness (initialize trackers from different frames) and spatial robustness (initialize trackers with varying degrees of added spatial error). The performance was compared according to the Area Under Curve (AUC) score of the two trade-off plots, i.e. success ratio versus overlap threshold and success ratio versus center error threshold. Later, Li [22] constructed an even larger RGB dataset NUS-PRO, featuring 365 videos of pedestrians and rigid objects. They followed the evaluation metrics used in [23, 129]. For RGB-D tracking, Song et al. [20] generated a dataset with 100 sequences. However, a common problem of these large datasets is that their attributes are *per-sequence* annotated rather than *per-frame* annotated, and all attributes are assigned based on the intuition of human annotators. This makes it hard to accurately categorise the tracker performance within different attributes. For example, per-sequence annotated attributes (e.g. "occlusion") do not last throughout the entire sequence [21]. It is preferable to evaluate each tracker with respect to attributes annotated on a per-frame basis, for a more accurate and meaningful analysis. Note that constructing a very large dataset does not guarantee diversity in its visual attributes, while it significantly slows down the

Table 2.2: Comparison of benchmark datasets. BL: binary label (human intuition based annotation).

Type	Name	Num.	Attributes	
			Unit	Tag
RGB	ALOV++	314	N/A	N/A
	VTB1.0	50	per-sequence	BL
	VTB2.0	100	per-sequence	BL
	NUS-PRO	365	per-sequence	BL
	VOT2013	16	per-frame	BL
	VOT2014	25	per-frame	BL
	VOT2015	70	per-frame	BL
Infrared	VOT2015TIR	20	per-frame	BL
RGB-D	PTB	100	per-sequence	BL

process of evaluation (by increasing the computational burden of each tracker trial). A better approach is to annotate each sequence with the visual attributes occurring in that sequence, and then perform clustering to reduce the size of the dataset, while still maintaining its diversity [130]. Based on this approach, Kristan et al. [3] organised the Visual Object Tracking (VOT) workshop associated with the ICCV/ECCV conferences. Note that based on the previous proposed evaluation protocol [23, 129, 20, 22], if a tracker fails early in a sequence, it will show a low success ratio, but if it fails late in a sequence, it may show high success ratio. Therefore, the VOT evaluation protocol re-initializes trackers when failure is detected, which avoids the sequential position of failure will eventually damage the evaluation among different trackers. However, their work annotated all attributes using human annotators which almost unavoidably constains some subjective errors. We summarize some recent benchmark datasets in table 2.2.

Despite the availability of several high quality benchmark datasets for RGB tracker evaluation [3, 21, 22, 23], those available for RGB-D tracker evaluation are far more limited, viewed in table 2.2. The benchmark dataset PTB, created by Song et al. [20], contains 100 sequences. However, we observe many sequences where the RGB and depth image pairs are significantly un-synchronised, shown in figure 2.5. This can cause ambiguities in annotated ground-truth, and performance evaluation. Therefore, this thesis presents a new RGB-D dataset with correctly synchronised RGB and D channels to test our approach in a consistent and unbiased way.

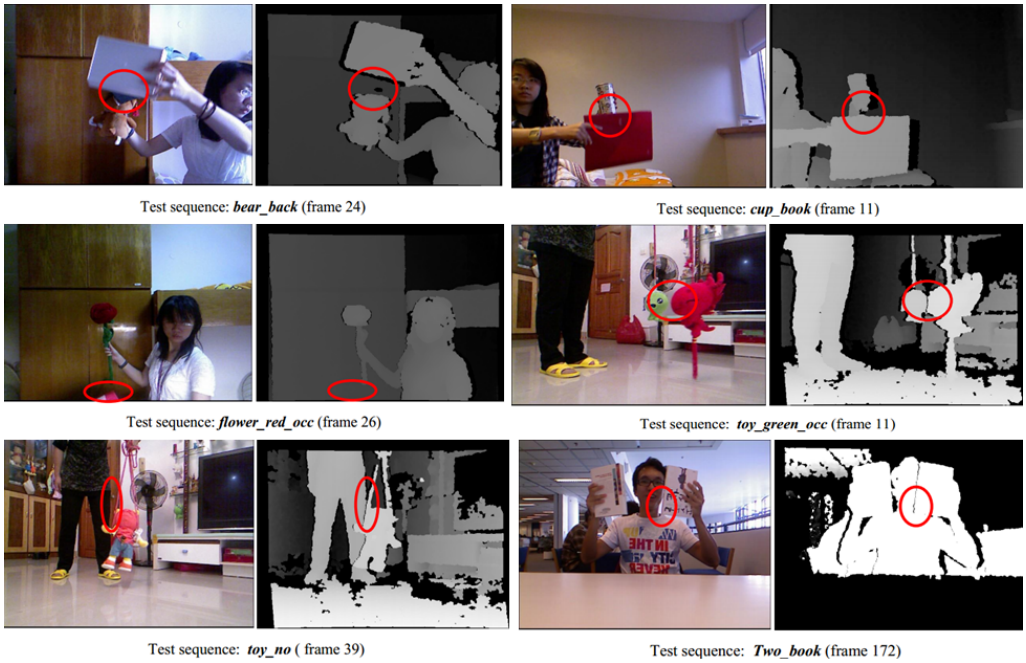

Figure 2.5: Some examples from Princeton RGB-D benchmark tracking dataset [20] which clearly show that RGB images and depth images are not synchronized.

2.4.2. Bias analysis.

Torralba et al [131] has proposed a fundamental question about "whether the datasets are measuring the right thing". Unlike datasets in machine learning, where the dataset *is* the world, computer vision datasets are supposed to *represent* the world [131]. Therefore, the bias inside the dataset can significantly affect the evaluation results. Good tracking performance might just indicate that the algorithm better fits the dataset bias. Ponce et al [132] pointed out that (i) some algorithms exploit some restrictions, but will fail when the restrictions do not apply; and, related to this, (ii) the images are not sufficiently challenging for the benefits of more sophisticated algorithms (e.g., scale invariance) to make a difference. Note that over half the PTB dataset [20] is devoted to pedestrian tracking and the camera is almost always stationary, which can bias the results of evaluating generic target trackers, see figure 2.6. To test a category-free tracker, it is important to build a dataset consisting of various object categories. Additionally, since motion of the RGB-D camera often causes significant depth variation, it is essential to evaluate how the tracker performs with camera motion.

Therefore, it is important to measure the dataset bias somehow to reduce difference between

Figure 2.6: Different images representing different testing sequences, which actually captured the same or very similar scenes. A large dataset (100 test sequences) does not have a meaningful impact on diversity if many of those sequences are the same or very similar.

the datasets and the real world. Torralba et al [131] suggested the potential sources of the bias are selection bias, capture bias and negative bias. Khosla et al [133] proposed a discriminative framework that directly exploits dataset bias, where the model learns two sets of weights: (1) bias vectors associated with each individual dataset, and (2) visual world weights that are common to all datasets, which are learned by undoing the associated bias from each dataset. Based on the study of dataset bias, we establish some criteria that could better guide the unbiased dataset construction, which is presented in this thesis (chapter 4).

CHAPTER 3

RGB TRACKER

This chapter proposes a new single target tracker for 2D RGB image sequences, under various kinds of difficult tracking conditions. The proposed method achieves state-of-the-art robustness by addressing the problems in three fundamental areas: target representation, matching methodology, and model adaptation [1]. The target is *represented* at three different layers: pixel layer, local-parts layer and target (bounding box) layer. A novel "Adaptive Clustered Decision Tree" method is proposed to enable robust *matching* at the part layer between successive frames, which dynamically selects the minimum combination of necessary features, thereby providing robustness with computational efficiency. The target model is *updated* through a closed loop with cross-constraints between different model layers. We test the tracker using two different tracking benchmarks [2, 3], based on two different test methodologies, and show it to be more robust than all other state-of-the-art methods from those tracking challenges while also demonstrating competitive tracking precision.

This chapter is organized as follows. The proposed method is described in section 3.1, where the model representation is first introduced, followed by the work of model matching and adaptation. Section 3.2 evaluates the proposed method using two benchmark datasets and demonstrates state-of-the-art performance. Additionally, the contribution of each key component of the tracker to overall performance is also evaluated. The chapter is concluded in section 3.3.

3.1 Proposed method

Following the conventions of [3] and [23], our method initialises its target model using only a given bounding box in the first frame, without any prior information about the target category.

3.1.1 Multi-layer target representation

The target is modelled hierarchically at three different layers: the pixel (bottom) layer, parts (middle layer, constructed from super-pixel) layer, and the target (top or overall representation) layer [1], see figure 3.1. Following the logic of the model construction and the order of model updating, this section first introduces the middle layer model, since the initial features are extracted from this layer. Later, we introduce the top layer followed by the bottom layer.

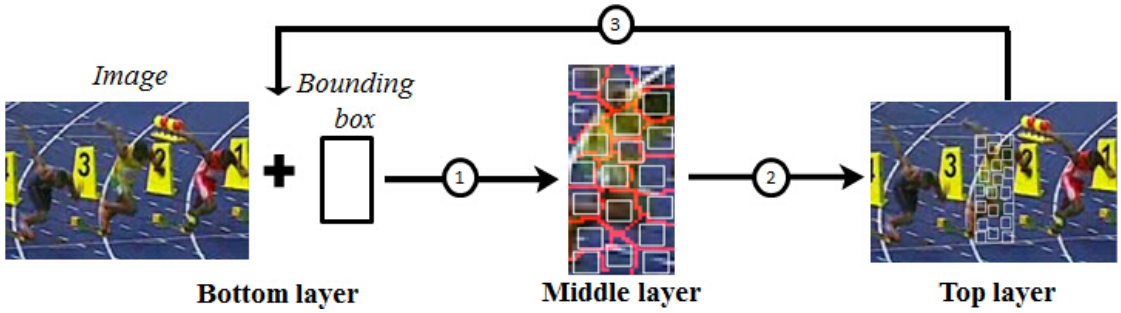

Figure 3.1: 1-use top layer propagation to find a candidate image region, and segment it into super-pixels; 2-match middle layer parts, update old parts, learn new parts, use updated parts to relearn top layer; 3-use new top layer to assign likelihoods to bottom layer pixels in the new frame.

3.1.1.A. Notation

We precede the model descriptions with a brief overview of notation. In principle, our method can be used with any combination of features:

$$\mathbf{F}_k = \{\mathbf{f}_k^1, ..., \mathbf{f}_k^i, ..., \mathbf{f}_k^{N_f}\} \tag{3.1}$$

where $\mathbf{F}_k$ denotes a set of N_f feature modalities exploited by the tracker at the kth video frame. Note that the number of feature modalities in each of the target layers (explained later) may be

different, and $\mathbf{f}_k^i$ denotes a particular kind of feature, provided that all such features, $\mathbf{f}_k^i$, satisfy the following conditions.

Firstly, it should be possible to associate any image pixel with a feature value:

$$\mathbf{I}_k(x) \mapsto \mathbf{f}_k^{i(x)} \tag{3.2}$$

where $\mathbf{I}_k(x)$ denotes the pixel at location x in the kth image, $\mathbf{I}_k$. Clearly, certain kinds of features, e.g. texture, must be calculated from a region surrounding an individual pixel, but such feature values can still be associated with the central pixel's location.

Secondly, in the ith feature modality, the set of feature values $\mathbf{f}_k^{i(x \in \boldsymbol{R}_k^q)}$ of any image region, $\boldsymbol{R}_k^q$, can be used to create a model:

$$\boldsymbol{R}_k^q \mapsto \boldsymbol{M}_k^{i(q)} \equiv \boldsymbol{M}_k^{i(x \in \boldsymbol{R}_k^q)} \tag{3.3}$$

where $\boldsymbol{M}_k^{i(q)}$ is a model of the qth region's pixels' feature values, which could for example be a probability distribution or some other kind of collective representation.

Thirdly, for the ith feature modality, a suitable likelihood function $\hat{L}_k^{i(p,q)}$ exists for comparing the similarity of any pair of models, $\boldsymbol{M}_k^{i(p)}, \boldsymbol{M}_k^{i(q)}$, representing any pair of image regions, $\boldsymbol{R}_k^p$ and $\boldsymbol{R}_k^q$:

$$\{\boldsymbol{M}_k^{i(p)}, \boldsymbol{M}_k^{i(q)}\} \mapsto \hat{L}_k^{i(p,q)} \tag{3.4}$$

Where such feature models are in the form of distributions, then corresponding likelihood functions are typically based on divergence measures, such as [77, 134, 135, 79]. However, in general, other kinds of useful likelihood functions can be formed for other kinds of models.

Fourthly, for each kind of feature model $\boldsymbol{M}_k^{i(q)}$, another likelihood function $\tilde{L}^{i(x,q)}$ exists for evaluating the consistency of any pixel's feature value, $\mathbf{f}_k^{i(x)}$, with the model. Intuitively, we can regard the set of such likelihoods:

$$\tilde{\boldsymbol{L}}_k^{(x,q)} \equiv \{\tilde{L}_k^{1(x,q)} \ldots \tilde{L}_k^{i(x,q)} \ldots \tilde{L}_k^{N_f(x,q)}\} \tag{3.5}$$

as the set of opinions of each feature modality as to the likelihood of such a pixel value belonging to the part of the target object projected as the qth image region, $\boldsymbol{R}_k^q$, in the kth frame. In cases where the feature models are in the form of distributions, then such likelihoods may be conveniently posed in the form of conditional probabilities.

3.1.1.B. Middle layer target representation

The middle layer consists of a set of feature models (in the sense of equation 3.3), extracted from the image regions corresponding to parts of the target, which we detect as super-pixels. Using super-pixels as the basis for middle layer model extraction offers several advantages over randomly selecting square "patches" as in previous methods [6, 136]. Firstly, a super-pixel is much more likely to correspond to a distinct and homogeneous part of the target. In contrast, randomly (or uniformly) selected patches are likely to contain pixels from two or more dissimilar (e.g. in terms of colour) parts of the target, which can lead to matching ambiguity. Secondly, when patches are randomly (or uniformly) selected from an initial bounding box, many patches are likely to contain information about both target and background pixels, and this also can negatively impact tracking performance. In contrast, due to the homogeneous nature of super-pixels, the pixel models extracted from these regions are much more likely to include either purely target pixels, or purely background pixels. Once tracking begins, those super-pixels which erroneously correspond to background regions are rapidly detected and eliminated from the middle layer target representation, leaving only those super-pixels which truly belong to target parts [1]. We use the Simple Linear Iterative Clustering (SLIC) super-pixel segmentation method [137], because it offers the following advantages: the number of super-pixels can be known in advance; the super-pixels have uniform size; the compactness can be defined, which enables us to use squares representing the superpixels; and the algorithm has high computational efficiency. Then, the middle layer target representation, $\zeta_k^{\texttt{mid}}$, consists of $N_{\texttt{mid}}$ feature models (of the form equation 3.3), for each of N_s super-pixels, representing N_s target parts, denoted as:

$$\zeta_k^{\mathtt{mid}} = \{\{\boldsymbol{M}_k^{1_{\mathtt{mid}}(1)} ... \boldsymbol{M}_k^{i_{\mathtt{mid}}(1)} ... \boldsymbol{M}_k^{N_{\mathtt{mid}}(1)}\},$$
$$..., \{\boldsymbol{M}_k^{1_{\mathtt{mid}}(q)} ... \boldsymbol{M}_k^{i_{\mathtt{mid}}(q)} ... \boldsymbol{M}_k^{N_{\mathtt{mid}}(q)}\},$$
$$..., \{\boldsymbol{M}_k^{1_{\mathtt{mid}}(N_s)} ... \boldsymbol{M}_k^{i_{\mathtt{mid}}(N_s)} ... \boldsymbol{M}_k^{N_{\mathtt{mid}}(N_s)}\}\} \quad (3.6)$$

where $\boldsymbol{M}_k^{i_{\mathtt{mid}}(q)}$ is a feature model in the ith feature modality for the qth super-pixel, representing the qth target part, at frame k.

3.1.1.C. Top layer target representation

The top layer of the target representation is denoted as $\zeta_k^{\mathtt{top}}$ which includes overall information about the target's bounding box region, $\boldsymbol{R}_k^T$, and also a ring-shaped local contextual region, $\boldsymbol{R}_k^C$, which surrounds the target region (see figure 3.2), in the form of a set of models for each region in each feature modality:

$$\zeta_k^{\mathtt{top}} = \{\boldsymbol{M}_k^{i_{\mathtt{top}}(T)}, \boldsymbol{M}_k^{i_{\mathtt{top}}(C)}\}_{i_{\mathtt{top}}=1...N_{\mathtt{top}}} \quad (3.7)$$

where T and C indicate a target region and a contextual region, respectively. The top layer target models, $\{\boldsymbol{M}_k^{1_{\mathtt{top}}(T)} ... \boldsymbol{M}_k^{i_{\mathtt{top}}(T)} ... \boldsymbol{M}_k^{N_{\mathtt{top}}(T)}\}$, are computed from the middle layer target parts models as:

$$\boldsymbol{M}_k^{i_{\mathtt{top}}(T)} = \frac{1}{N_s} \sum_{q=1}^{N_s} \boldsymbol{M}_k^{i_{\mathtt{mid}}(q)} \quad (3.8)$$

where $\{\boldsymbol{M}_k^{i_{\mathtt{mid}}(q)}\}_{q=1...N_s}$ are the middle layer feature models for each of N_s target parts, as defined in equation 3.6. The top layer local contextual models, $\{\boldsymbol{M}_k^{1_{\mathtt{top}}(C)} ... \boldsymbol{M}_k^{i_{\mathtt{top}}(C)} ... \boldsymbol{M}_k^{N_{\mathtt{top}}(C)}\}$, are computed directly from the pixels occupying the ring-shaped local contextual region surrounding the target:

$$\boldsymbol{M}_k^{i_{\mathtt{top}}(C)} \equiv \boldsymbol{M}_k^{i_{\mathtt{top}}(x \in \boldsymbol{R}_k^C)} \quad (3.9)$$

where $\boldsymbol{R}_k^C$ is determined by enlarging the target's bounding box area by multiplying both width and height by a scale factor, λ (value 1.2) , as shown in figure 3.2.

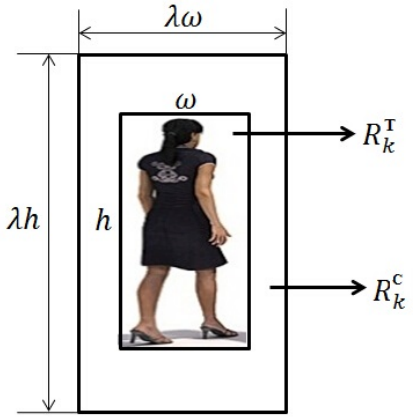

Figure 3.2: Target, $\boldsymbol{R}_k^T$, and ring-shaped local contextual, $\boldsymbol{R}_k^C$, region. h and ω is the height and width of the bounding box, respectively. The scaling factor is λ.

3.1.1.D. Bottom layer target representation

The bottom layer representation, $\zeta_k^{\mathtt{bot}}$, comprises individual pixels, each associated with a set of likelihoods, according to each feature modality, that the pixel belongs to the target object:

$$\zeta_k^{\mathtt{bot}} = \{l_k^{1_{bot}(x)}..., l_k^{i_{bot}(x)}..., l_k^{N_{bot}(x)}\}_{x\in(\boldsymbol{R}_k^T \cup \boldsymbol{R}_k^C)} \tag{3.10}$$

where $l_k^{i_{bot}(x)}$ denotes the discriminative likelihood of a pixel x belonging to the target with respect to the i_{bot}th feature modality, computed as:

$$l_k^{i_{bot}(x)} = \frac{\lambda^{\mathtt{top}} \tilde{L}_k^{i_{top}(x,T)}}{\lambda^{\mathtt{top}} \tilde{L}_k^{i_{top}(x,T)} + (1-\lambda^{\mathtt{top}}) \tilde{L}_k^{i_{top}(x,C)}} \tag{3.11}$$

where $\tilde{L}_k^{i_{top}(x,T)}$ denotes the likelihood of a pixel value, $\mathbf{I}_k(x)$, with respect to the top layer model, $\boldsymbol{M}_k^{i_{\mathtt{top}}(T)}$, of the target region, $\boldsymbol{R}_k^T$; $\tilde{L}_k^{i_{top}(x,C)}$ denotes the likelihood of the pixel value with respect to the top layer model, $\boldsymbol{M}_k^{i_{\mathtt{top}}(C)}$, of the target's local contextual region, $\boldsymbol{R}_k^C$; $\lambda^{\mathtt{top}}$ is the expected ratio of the top layer target area to the top layer contextual area. For simplification, it is computed as: $\lambda^{\mathtt{top}} = (\lambda - 1)/2$, approximately.

3.1.2 Tracker propagation and matching

An overall schematic for the tracker is shown in figure 3.3. The tracker is first propagated by information matching at the top layer, which generates a candidate image region for the middle layer. Next, this candidate region is segmented, into equally sized super-pixels. We next propose a continuously-adaptive clustered decision tree method, which efficiently associates middle layer target parts (learned from super-pixels in the previous frame), onto candidate super-pixels in the new frame, using the minimum number of features for each part [1]. Lastly, to cope with target deformation and appearance changes, well matched parts are adaptively updated, while parts which cannot be satisfactorily matched are deemed to no longer be visible (e.g. due to occlusion) and are temporarily "switched off", but stored in memory, from the mid-layer model. The severely drifting parts, defined as those which drift far from the main body of the target, will be replaced by new candidates. In this case, another decision tree is then used to choose the best super-pixels for creating new middle layer parts to replace those drifting parts [137].

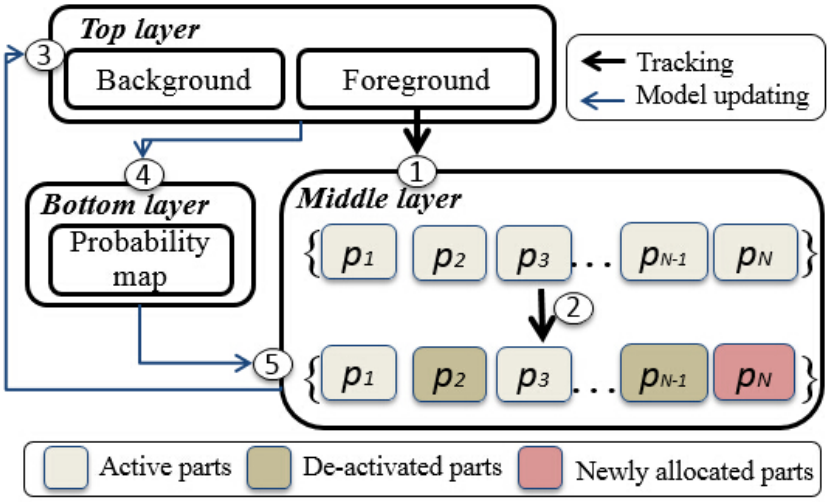

Figure 3.3: Block diagram of the proposed tracking process. 1- tracker propagation by the top layer target information matching; 2- middle layer (target part) matching with clustered decision tree; 3- update the top layer model; 4- feed back the bottom layer (pixel) feature; 5- re-sample drifting parts at the middle layer model [1].

3.1.2.A. Top layer propagation

In the top layer, we obtain a set of uniformly distributed samples $\{\hat{\boldsymbol{R}}_k^T(j)\}_{j=1...N_R}$ to represent the posterior density function of the target, with associated weights.

$$\hat{\boldsymbol{R}}_k^T(j) = \boldsymbol{R}_{k-1}^T + \boldsymbol{V}_k^T(j) \tag{3.12}$$

where $\hat{\boldsymbol{R}}_k^T(j)$ represents the hypothetical state (region) of the target computed from the previous tracker state $\boldsymbol{R}_{k-1}^T$ with a uniformly assigned moving velocity $\boldsymbol{V}_k^T(j)$.

We follow the conventional methods, e.g. [64], to estimate the overall target position using the expectation operator over the set of samples, which can be denoted as:

$$\boldsymbol{R}_k^T = \sum_{j=1}^{N_R} \hat{\boldsymbol{R}}_k^T(j) p(\hat{\boldsymbol{R}}_k^T(j)|\boldsymbol{R}_{k-1}^T) \tag{3.13}$$

where $\hat{\boldsymbol{R}}_k^T(j)$ represents the hypothetical state (region) of the target, referred to jth sample at k frame, with associated normalized weight $p(\hat{\boldsymbol{R}}_k^T(j)|\boldsymbol{R}_{k-1}^T)$ such that $\sum_{j=1}^{N_R} p(\hat{\boldsymbol{R}}_k^T(j)|\boldsymbol{R}_{k-1}^T) = 1$ and N_R is the number of the samples. Here, the associated weights of the underlying samples are computed from the similarity between target top layer models and candidate regions' models. For one specific model i of the sample j, the corresponding weight can be denoted as:

$$p(\hat{\boldsymbol{R}}_k^{i_{\text{top}}}(j)|\boldsymbol{R}_{k-1}^{i_{\text{top}}}) = exp\{\frac{\hat{L}_k^{i_{\text{top}}}(\hat{\boldsymbol{M}}_k^{i_{\text{top}}(T)}(j), \boldsymbol{M}_{k-1}^{i_{\text{top}}(T)})^2 - 1}{\sigma_{i_{\text{top}}}^2}\} \tag{3.14}$$

where $\hat{\boldsymbol{M}}_k^{i_{\text{top}}(T)}(j)$ is the ith observed target model of sample j in top layer, while $\boldsymbol{M}_{k-1}^{i_{\text{top}}(T)}$ is the corresponding reference model. $\sigma_{i_{\text{top}}}$ is the standard deviation and $\hat{L}_k^{i_{\text{top}}}$ is the same likelihood function in equation 3.4. Our later proposed adaptively clustered decision trees could be used for fusing the weights computed from different models of the same sample.

3.1.2.B. Middle layer matching with adaptive clustered decision tree

After propagating the top layer to give a candidate bounding box region, this is further enlarged to form a broader search region, which we then segment into super-pixels [137], which provide

a pool of candidate image regions for matching to middle layer target parts, using our proposed adaptive clustered decision tree method. Typically, the structures of decision trees are generated offline during training, e.g. [75, 86], but such static trees would be unsuitable for tracking targets with dynamically changing appearances, against changing background scenes. Instead, we propose an adaptive tree structure, which is relearned online for each new image, by explicitly considering contextual information.

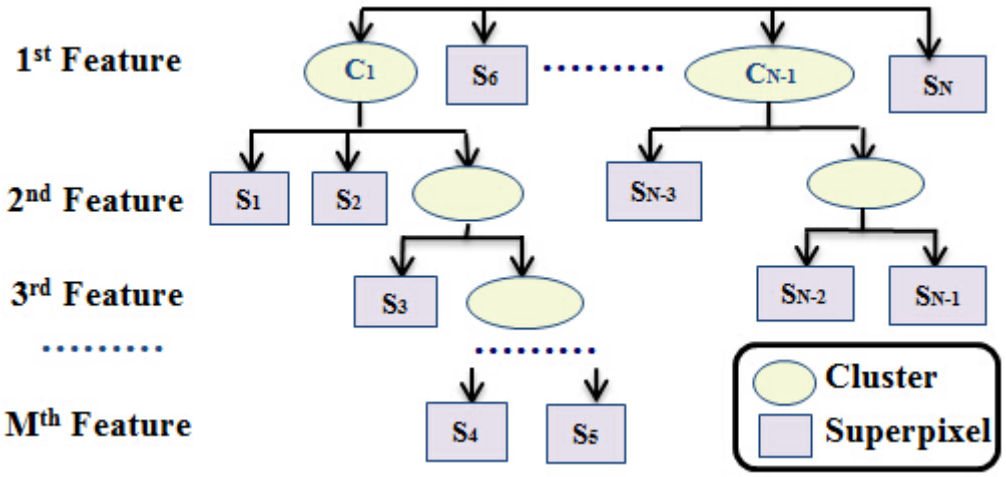

Figure 3.4: Clustered decision tree. Each tree-level represents a feature modality (e.g. color, motion etc.). The feature values of superpixels form leaves on a tree-level. If any two leaves are sufficiently similar in feature value then they are merged into a cluster. A new tree-level (using the next feature modality) is then used to try to disambiguate the members of such clusters. The tree continues growing (by adding more tree-levels of more features), until all target parts have been assigned to a unique choice of superpixel. Any remaining unmatched parts are assumed to have become occluded and are temporarily switched off [1].

The proposed adaptive clustered decision tree method is illustrated in figure 3.4. The objective of the decision tree is to find the corresponding super-pixel which best matches each middle layer target part, while adaptively selecting the minimum combination of features needed to do this robustly in each frame [1]. The first tree-level is initialised by selecting a primary feature from the set of all features, and labelling all candidates (constructed from super-pixels) as individual leaves. Next, all leaves are compared for similarity in the primary feature modality. Similar candidates are grouped into clusters according to a label map:

$$\mathfrak{C}_k^{i_{\mathtt{mid}}(p_s,q_s)} = \begin{cases} 1, & \hat{L}_k^{i_{\mathtt{mid}}(p_s,q_s)} > T_k^{i_{\mathtt{mid}}} \\ 0, & others \end{cases} \tag{3.15}$$

where $\hat{L}_k^{i_{\mathtt{mid}}(p_s,q_s)}$ is the similarity measurement for $i_{\mathtt{mid}}$th feature model between candidates p_s and q_s and $T_k^{i_{\mathtt{mid}}}$ is a clustering threshold (relearned online as described later). The clustering process is conducted at each tree-level except at the bottom tree-level.

Next, each terminating leaf and cluster is compared against the part in the middle layer target model for which a match is sought.

$$\hat{p_s} = \arg\max \ \hat{L}_k^{i_{mid}(p_s,q_t)} \quad s.t. \ \hat{L}_k^{i_{mid}(p_s,q_t)} > 0 \tag{3.16}$$

where q_t indicates the target part, p_s is the candidate leaf or cluster. When p_s represents a cluster, its associated feature model is denoted by the mean value of the constituent candidates. $\hat{L}_k^{i_{mid}}$ is the similarity score of feature model i_{mid}.

If any candidate super-pixel is both i) distinct enough from others that it forms its own leaf and does not lie inside a cluster (e.g. S_6 or S_N in figure 3.4) and ii) matches the target part with a relatively high similarity score (above the threshold used in equation 3.16), then the decision tree ceases splitting and the middle layer target part is labelled as matching that candidate local region. However, if the best match p_s for middle layer part q_t is one of the clusters, then this suggests that the primary feature is not sufficiently discriminating to enable q_t to be matched to a unique super-pixel. We therefore grow a new tree-level, based on a second feature, i.e. the cluster grows a new child leaf for each of its constituent candidates (e.g. S_1 or S_2 on second row in figure 3.4), where each leaf in the new tree-level is modeled using the secondary feature modality. This tree extending procedure continues (by adding additional tree-levels for additional features) until a unique patch-to-leaf correspondence is found, e.g. S_4 or S_{N-1} in figure 3.4. Once all the middle layer parts have been matched to new locations in the current image, the boundary of their spatial extent is used to output a new bounding box position, and the top layer target models are updated accordingly [1]. The proposed matching algorithm is a

greedy method, and the clustered decision trees need to be recomputed on each frame.

Occasionally, some parts in the model will fail to find a strongly matching candidate, even after exhausting all possible image features (corresponding to all possible tree levels). In such cases, it is inferred that the part is no longer visible and it is temporarily switched off. Other methods in the literature (e.g. [6]) remove unmatched patches, and thus lose parts of the model during occlusions, which cannot later be recovered. In contrast, our proposed approach of temporarily switching off the unmatched parts provides a powerful tracking memory that automatically handles occlusion situations without the need for special additional occlusion routines [1].

Note that the threshold, $T_k^{i_{mid}}$, has an important and useful effect on the extent to which each feature is used in the mid-layer patch matching procedure. High values of $T_k^{i_{mid}}$ reduce the amount of clustering on the respective tree-level, ensuring that most candidate local regions will be represented as individual leaves, i.e. the algorithm will distinguish most candidates using this feature alone [1]. In contrast, low values of $T_k^{i_{mid}}$ make it likely that this tree-level will grow many clusters, with the effect that the overall algorithm will make less use of this particular feature for matching, and will rely on later features (lower levels on the tree) to provide discrimination. In other words, the choice of $T_k^{i_{mid}}$ can actually be regarded as a measure of confidence in the discriminating ability of a feature.

Consequently, we would like to set high values of $T_k^{i_{mid}}$ for those features that are highly discriminatory in the current frame, and low values of $T_k^{i_{mid}}$ for poorly performing features (e.g. those modalities for which the target is camouflaged against the context in the present frame). We therefore propose a method by which $T_k^{i_{mid}}$ is continuously relearned for each feature at each frame, based on evaluating the feature's discriminating ability relative to the current contextual information. $T_k^{i_{mid}}$ is computed online as:

$$T_k^{i_{mid}} = \hat{L}_k^{i_{top}(T,C)} \tag{3.17}$$

where $\hat{L}_k^{i_{top}(T,C)}$ indicates the similarity score of models $\boldsymbol{M}_k^{i_{top}(T)}$, $\boldsymbol{M}_k^{i_{top}(C)}$ which are the top layer target and ring-shape models defined in equation 3.7, and equation 3.17 is therefore a measure of similarity between the target and the local context (i.e. a measure of camouflage) in

the respective feature modality.

The advantages of the adaptive decision tree are: firstly, its structure is adaptively generated online, which overcomes the over-fitting of offline generated classifiers; secondly, rather than using all features in all frames, the adaptive tree efficiently exploits only the minimum number of necessary features, ensuring sufficient robustness for minimal computational cost; thirdly, the decision tree adaptively weights in favour of the most discriminating features, dynamically responding to changing amounts of camouflage in different feature modalities.

3.1.2.C. Model updating

For robust tracking, it is necessary to continuously update the target model as it changes its appearance with time. The proposed tracker does this via two mechanisms: adapting the old middle and top layer target models, and adding new models of new middle layer target parts (derived from super-pixels) [1].

- *Learning new target parts*

At each frame, we examine all the matched middle layer target part models and detect those which are drifting (moving too far from the target centroid) [136]. The algorithm first searches the maximum non-zero interval of the marginal distribution according to the matched parts. The region outside the maximum non-zero interval becomes regarded as a potential drifting region and is then chosen as a candidate for additional drift checking according to the following criteria: 1) only a small group of parts, which are separated from the majority cluster; 2) the distribution density of the drifting patches should be very small [136]. To replace those middle layer target parts that were detected as drifting, we select the "available" candidate local regions, i.e. those which are not yet matched to a target part, and rank them in order of their likelihood of representing target parts. A second kind of adaptive clustered decision tree is used to perform this ranking as follows.

We begin by using the primary feature modality to generate the leaves of the first level of the decision tree, according to a ranked (descending order) list of unmatched candidate likelihoods

$L_k^{i_{mid}(p_s)}$ (explained later in the experiment section). We then cluster those candidates (super-pixels) which share similar likelihoods, according to:

$$\mathfrak{C}_k^{i_{mid}(p_s,q_s)} = \begin{cases} 1, \|L_k^{i_{mid}(p_s)} - L_k^{i_{mid}(q_s)}\| < \lambda_{rank}\sigma_{all} \\ 0, \quad others \end{cases} \tag{3.18}$$

where λ_{rank} is a pre-defined parameter while σ_{all} is the standard deviation of all candidates' expected likelihoods. p_s and q_s is the index of candidate local regions.

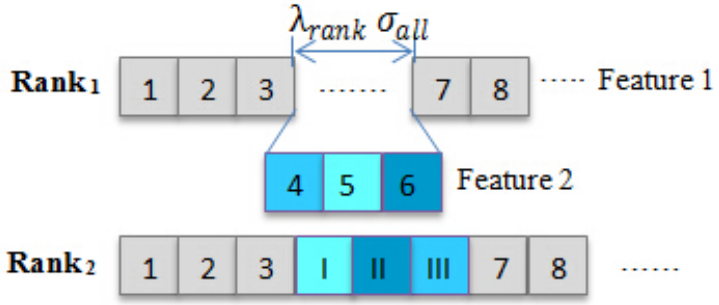

Figure 3.5: Rank the candidate local regions with clustered decision trees (descending order). $Rank_1$ ranks the candidates only using primary feature (equivalent to the first tree-layer); $Rank_2$ re-ranks the candidates by adding additional feature (equivalent to the second tree-layer).

Once again, a cluster suggests that the feature for this tree-level is not sufficiently discriminating to achieve a robust ranking [1]. Therefore, a secondary feature is chosen and used to rank all constituent candidates within the cluster, forming a second tree-level. The tree is grown (by adding successive tree-levels, using successive features), until a unique ranking has been assigned to a sufficient number of available candidate local regions, in order to replace those target parts (which were removed due to being identified as drifting) by the candidates with highest rank.

- *Updating old target parts*

For those middle layer target parts that were matched strongly onto candidates in the new frame, the feature models are updated according to new observations. Note that any kind of target model relearning is potentially dangerous, since even small tracking errors can easily cause background pixels to be learned into the target model, leading to instability with exponentially

increasing errors. Early colour particle filter work [64], and recent state-of-the-art patch-based methods [6], perform model relearning at a fixed update speed. In contrast, we continuously re-compute individual update speeds for each middle layer target part at each frame. Our premise is that parts can be relearned rapidly when there is a high confidence in their matching, whereas the relearning rate should be reduced under conditions of uncertainty [1]. We therefore update each patch, using a continuously relearned parameter, $\mu_k^{i_{mid}(p_t)}$, according to the following update rule:

$$\boldsymbol{M}_k^{i_{mid}(p_t)} = (1 - \mu_k^{i_{mid}(p_t)})\boldsymbol{M}_{k-1}^{i_{mid}(p_t)} + \mu_k^{i_{mid}(p_t)}\boldsymbol{M}_{obs}^{i_{mid}(p_t)} \tag{3.19}$$

$$\mu_k^{i_{mid}(p_t)} = \hat{L}(\boldsymbol{M}_{k-1}^{i_{mid}(p_t)}, \boldsymbol{M}_{obs}^{i_{mid}(p_t)}) \tag{3.20}$$

where $\boldsymbol{M}_{k-1}^{i_{mid}(p_t)}$ is the model of the p_tth target part, in the ith feature modality, at frame $k-1$, while $\boldsymbol{M}_{obs}^{i_{mid}(p_t)}$ is the observed model of the corresponding matched super-pixel in the current frame. Again, $\hat{L}(.)$ is a similarity score for the ith feature modality, as defined in equation 3.4.

- *Updating the top layer target representation*

At each frame, once all middle layer target parts have been either switched off, updated, or replaced, then the top layer target model is updated according to equation 3.8, as described in section 3.1.1.C.

3.1.2.D. Handling occlusions

The proposed tracker utilises a memory which memorises the latest tracker state, including all middle layer target part models [1]. As described in section 3.1.2.B, partial occlusion is handled by temporarily switching off poorly matching middle layer parts, but retaining these in memory and reacquiring them once occluded target parts reappear in later video frames. The tracker is regarded as being in a special state of full occlusion if a large proportion of patches (defined by a threshold parameter) remain unmatched after the matching procedure of section 3.1.2.B:

$$\mathbb{O}_k = \begin{cases} 1, N_o/N_s > T_o \\ 0, \ \ others \end{cases} \tag{3.21}$$

where $\mathbb{O}_k$ is occlusion status (1. occlusion; 0. non-occlusion), N_o is the number of the occluded parts and N_s is the total number of all parts. T_o is the threshold to check the full occlusion status. In the full occlusion state, the target model updating (at all model layers) stops, and the top layer sample propagation scope $\boldsymbol{V}_k^T(j)$ in equation 3.12 is enlarged (twice) to handle an increased degree of uncertainty. After finding the best candidate region from the equation 3.13, the algorithm performs the procedure in section 3.1.2. The tracker returns to the normal (non-occlusion) state once a sufficient proportion of middle layer parts (similarly defined using equation 3.21) are matched strongly to candidate local regions (super-pixels).

3.2 Experimental results and analysis

In this section, we first test the performance of our tracker on sequences from the publicly available benchmark datasets VOT 2013, VOT 2014 and VTB 2013 benchmark datasets [3, 23], which together comprise 70 sequences in total. These datasets are currently considered state-of-the-art benchmarks in the 2D visual object tracking community. More details of the datasets can be found from the webpages of [3] and [2]. Note that the VOT and VTB benchmarks [3, 23], also represent two different evaluation methodologies. VTB [23] runs each tracker throughout each sequence without reinitialisation following failures while VOT [3] re-initialises trackers whenever a failure is detected. We show that the proposed tracker is significantly more robust than the state-of-the-art methods from both of those tracking challenges, while also offering competitive tracking precision.

Next, we carry out a decomposition analysis to understand which parts of the novel tracker contribute to this strong performance. The key elements of the tracker include i) the clustered decision tree, ii) the use of super-pixels to define middle-layer target parts or patches, iii) the use of an adaptive target re-learning rate, which re-learns faster during periods of high confidence,

and slows down target re-learning during periods of greater uncertainty. We decompose the performance by evaluating the tracker with and without each of these novel elements. The contribution of the clustered decision tree is evaluated, by replacing it with a more conventional parts matching method from the literature and comparing performance. The contribution of super-pixels for determining mid-layer target parts is evaluated by replacing it with a more conventional method from the literature (random assignments of target regions to be mid-layer parts). The contribution of the variable adaptive re-learning rate is evaluated by replacing it with a fixed (mid-range) re-learning rate (as in most of the comparable literature). In each case, all other parts of the algorithm are kept constant.

3.2.1 Implementation

The proposed adaptive clustered decision tree structure is designed to handle, in principle, arbitrarily many features in a robust and efficient manner. For proof of principle, we demonstrate the performance of the method on RGB images by implementing a tree with just two features, however this already delivers strongly competitive performance on benchmark test data [1].

For our primary (top of the tree) feature in the middle layer parts matching, we use simple pixel RGB colour values, i.e. $\mathbf{f}_k^{1_{mid}(x)}$ returns the RGB value of pixel $\mathbf{I}_k(x)$. For the primary feature model, $\boldsymbol{M}_k^{1_{mid}(p)}$, we use a simple colour histogram:

$$\boldsymbol{M}_k^{1_{mid}(p)}(b) = \sum_{x \in \boldsymbol{R}_k^p} \delta[b, \mathbf{f}_k^{1_{mid}(x)}]/N_{pixel} \tag{3.22}$$

where δ is the Kronecker delta function, b is the bin in the feature model and N_{pixel} is the number of the pixels inside the region $\boldsymbol{R}_k^p$.

For this colour feature, we use a likelihood function between target part q_t and candidate image region (superpixel) p_s, $\hat{L}_k^{1_{mid}(p_s,q_t)}$, computed from the Bhattacharyya coefficient [77]:

$$\hat{L}_k^{1_{mid}(p_s,q_t)} = \sum_{b=1}^{N_b} \sqrt{\boldsymbol{M}_k^{1_{mid}(p_s)}(b) * \boldsymbol{M}_k^{1_{mid}(q_t)}(b)} \tag{3.23}$$

where N_b is the number of the bin.

For the secondary feature, we imply a motion feature, where candidate local regions are assigned high matching likelihoods if they imply a small frame-to-frame motion for the part being matched, which can be denoted as:

$$\hat{L}_k^{2_{mid}(p_s,q_t)} = \|\boldsymbol{X}_k^{p_s} - \boldsymbol{X}_k^{q_t}\| \tag{3.24}$$

where $\boldsymbol{X}_k^{p_s}$ is the center of the candidate local region p_s while $\boldsymbol{X}_k^{q_t}$ is the center of target part q_t.

In the model updating (in section 3.1.2.C), to form the 1^{st} level of the trees for drifting parts replacement, we use the 1^{st} feature modality to calculate, for every unmatched candidate, the primary expected likelihood $L_k^{1_{mid}(p_s)}$ (in equation 3.18) of that candidate's constituent pixels belonging to the target, which is computed as:

$$\hat{L}_k^{1_{mid}(p_s)} = \frac{1}{N_{p_s}} \sum_{x \in \boldsymbol{R}_k^{p_s}} l_k^{1_{bot}(x)} \tag{3.25}$$

where $L_k^{1_{mid}(p_s)}$ is the expected likelihood of the p_sth candidate local region belonging to the target, N_{p_s} is the number of pixels in the p_sth candidate, and $l_k^{1_{bot}(x)}$ is as defined in equation 3.11.

For the secondary feature for updating, we utilize a dense feature to assign an additional likelihood to the clustered candidate local regions:

$$\hat{L}_k^{2_{mid}(p_s)} = eps(-\|\boldsymbol{X}_k^{p_s} - \boldsymbol{X}_k^{c}\|) \tag{3.26}$$

where $\boldsymbol{X}_k^{p_s}$ is the center of candidate local region p_s while $\boldsymbol{X}_k^{c}$ is the center of the matched parts.

The tracking algorithm has been implemented on an Intel Core i5-3570 CPU, using Matlab code (linked also to some C++ components). This unoptimised implementation achieves near-to-real-time performance of 8fps. The key parameters initialised in the first frame are listed in the table 3.1.

Table 3.1: Values of key algorithmic parameters

Section	Equation	Value
Initialization	$\lambda^{\texttt{bot}}$ in equation 3.11	0.1
Decision tree	λ_{rank} in equation 3.18	0.1
Occlusion	T_o in equation 3.21	40%

3.2.2 Performance evaluation on RGB benchmark data sets

We first evaluate our tracker using the ICCV2013 [23] and ECCV2014 [3] "VOT challenge" testbeds, which use the following methodology: stochastic trackers are run 15 times and results are averaged; whenever a tracking failure is detected (bounding box has zero overlap with ground truth), the tracker is re-initialised. Table 3.2 compares the performance of our tracker against the best 4 VOT trackers, out of around 30 trackers that those challenges evaluated. Results are shown in terms of robustness (total number of failure instances) and accuracy (percentage overlap between each tracker's output bounding box and the ground truth bounding box) averaged over all frames.

The accuracy at frame k is defined as a bounding box overlap φ_k, using the tracker-output bounding box $\boldsymbol{R}_k^T$ and ground-truth bounding box $\boldsymbol{R}_k^G$ in equation 3.27:

$$\varphi_k = \frac{|\boldsymbol{R}_k^T \cap \boldsymbol{R}_k^G|}{|\boldsymbol{R}_k^T \cup \boldsymbol{R}_k^G|} \tag{3.27}$$

where $\cap$ and $\cup$ represent the intersection and union of two regions and $|\bullet|$ is the region size measured by pixels number.

For robustness, the protocol sets a threshold for overlap to indicate tracking success. The number of correctly tracked frames is divided by the total number of frames, equation 3.28, for a more comparable evaluation.

$$P_\tau(T,G) = \frac{||\{k|\varphi_k > \tau\}_{k=1}^N||}{N} \tag{3.28}$$

where τ denotes the threshold of the overlap, and N is the number of total frames. A failure of one frame is identified when overlap (computed in equation 3.27) is below the defined threshold τ (zero in experiment). The normalised number of correctly tracked frame is used to represent

Table 3.2: Visual object tracking challenge results in 2013 (associated with ICCV) and 2014 (associated with ECCV) [3]: comparing against the corresponding best 4 trackers. Fail.: failure. Acc.: accuracy.

	VOT 2013 (16 sequences)				
	Ours	PLT13 [138]	LGT++ [136]	EDFT [139]	FoT [140]
Fail.	0	0	1.53	14	22
Acc.	0.59	0.58	0.57	0.58	0.63
	VOT 2014 (25 sequences)				
	Ours	PLT14 [138]	DGT [141]	DSST [142]	SAMF [143]
Fail.	1	4	25	29	32
Acc.	0.52	0.56	0.58	0.62	0.61

the robustness of a tracker.

In terms of robustness, our tracker has zero failures in VOT2013 and only one failure in VOT2014. The next best algorithm is PLT which also achieved zero failures in VOT2013. However, note that the version of PLT tested in VOT2013 used a fixed bounding box size. Therefore this algorithm was unable to adapt to targets which change their size during tracking (e.g. due to range changes). However, since most objects in most test sequences luckily stayed roughly the same size in the VOT2013 benchmark videos, this rigid size constraint helped the algorithm to achieve an (arguably artificially) high robustness score. For VOT2014, a different version of PLT was submitted, which did enable adaptation to changing target size. In this case, PLT's robustness worsened to four failures [1].

Note that the "accuracy" scores can be highly misleading. In the VOT testing methodology, ground truth is used to re-initialise trackers (with perfect accuracy) after every tracking failure. Therefore, trackers which fail very often will show high accuracy scores, even if they are not "good" trackers. Hence, the accuracy score is only meaningful when comparing two trackers which have the same robustness score. In VOT2013 our accuracy is better than the only other algorithm (PLT) which shares the same robustness, while in VOT2014 no other tracker was able to achieve the same robustness [1].

For more extensive comparison, we also combine the VOT test sequences with all those from the VTB 2013 tracking benchmark data set [23]. Using this 70-sequence dataset, we compared our method against the publicly available trackers which have shown strong performance in

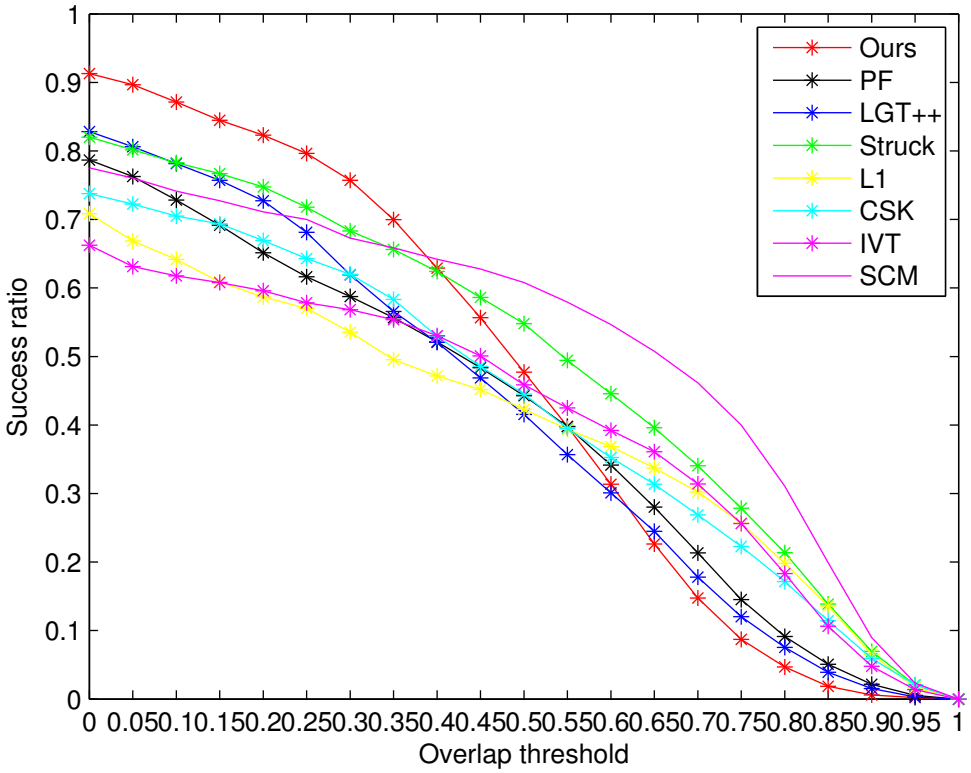

Figure 3.6: The success ratio versus overlap threshold curve in 70 sequences.

either the VOT or the VTB tracking challenges, namely: Struck [25], SCM [7], LGT++ [136], CSK [144], IVT [109], L1 [112], and PF [64]. Note that this experiment is complementary to the previous VOT testing, since the VTB benchmark system [23] uses a different evaluation methodology. In particular, since this dataset contains instances of full occlusions, the evaluation is conducted without re-initialization after tracking failures. We show the results as trade-off curves as suggested by [23].

As shown in figure 3.6, our tracker achieves significantly better robustness in accuracy ranges up to 0.4. We assert that robustness is the most important of these metrics: firstly, these methods are intended for highly deformable targets (e.g. people) for which it is hard to meaningfully interpret the "accuracy" of a rectangular bounding box which includes many non-target pixels even during good performance; secondly, provided that a tracker is robust, accuracy can always be further improved by incorporating additional prior knowledge about a specific target [142].

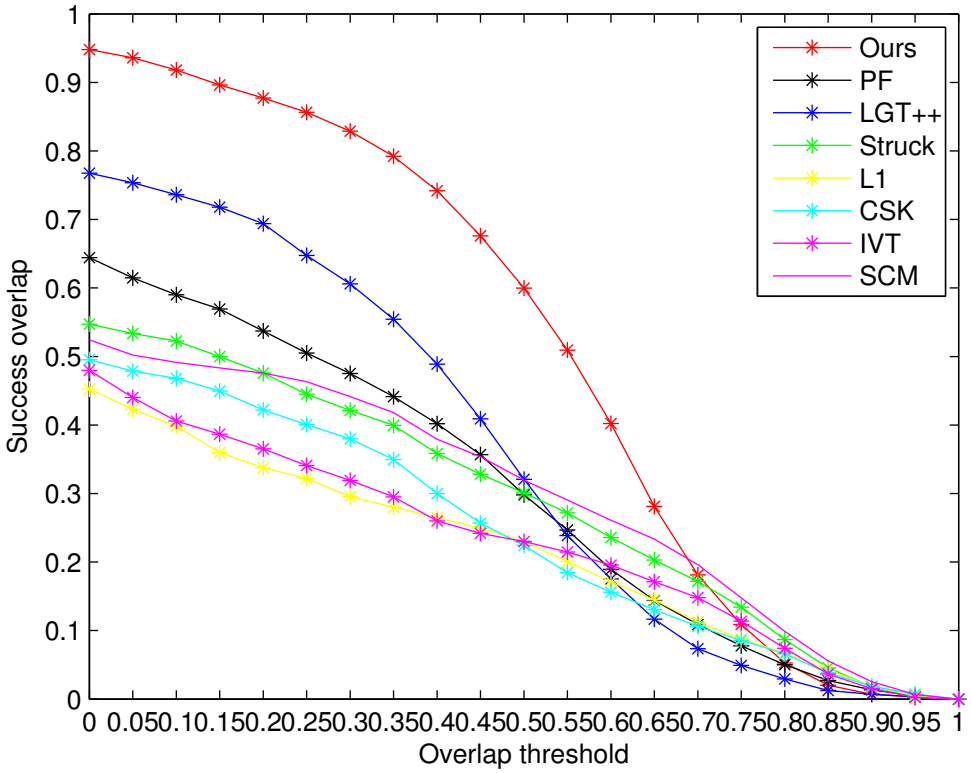

Figure 3.7: The success ratio versus overlap threshold curve in 19 sequences with deformation.

To further evaluate the performance, we also show the trade-off curves for those test videos identified in the benchmark challenges as containing the categories of: significant target deformations, figure 3.7; severe illumination changes, figure 3.8; and occlusions, figure 3.9. Our tracker significantly outperforms the other methods in tracking of highly deforming targets. We attribute this performance under deformation conditions to the flexibility of the clustered decision tree approach to online model relearning. The tracker also achieves competitive results in illumination change and occlusions scenarios. We attribute the strong performance under illumination changes to the robustness of the cross-constrained multi-layer target model. We attribute the results of the occlusion tests to the generality and adaptability of the proposed method. When a method is designed to be robust against dramatic target appearance and shape changes, it may not always be possible to distinguish between appearance changes and occlusions, hence our method sacrifices some accuracy in favour of robustness in such circumstances [1].

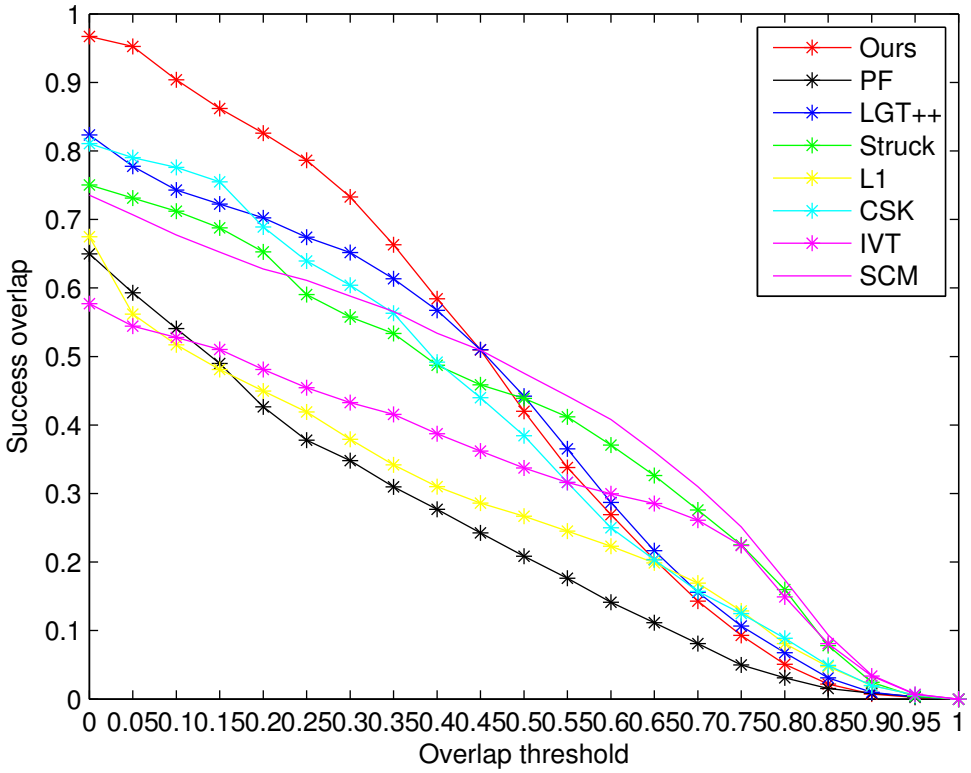

Figure 3.8: The success ratio versus overlap threshold curve in 18 sequences with illumination change.

Figure 3.10 shows examples of our method handling vidoes which feature extremely deforming targets (e.g. gymnast) and very strong clutter (e.g. diver).

3.2.3 Performance contributions of tracker sub-components

The previous section has shown that our proposed tracker performs extremely robustly in comparison to the best state-of-the-art trackers from the literature. However, our tracker relies on several key components which differ from other tracking methods. This section investigates the extent to which each of these components contributes to the overall strong performance of the tracker. To demonstrate the effect of each novel component, we evaluate performance while replacing that component with a conventional method from the literature, but keeping all other parts of the algorithm the same. Firstly, we remove our proposed use of super-pixels for determining mid-layer target parts, and replace it with the well-known method of [6], which selects

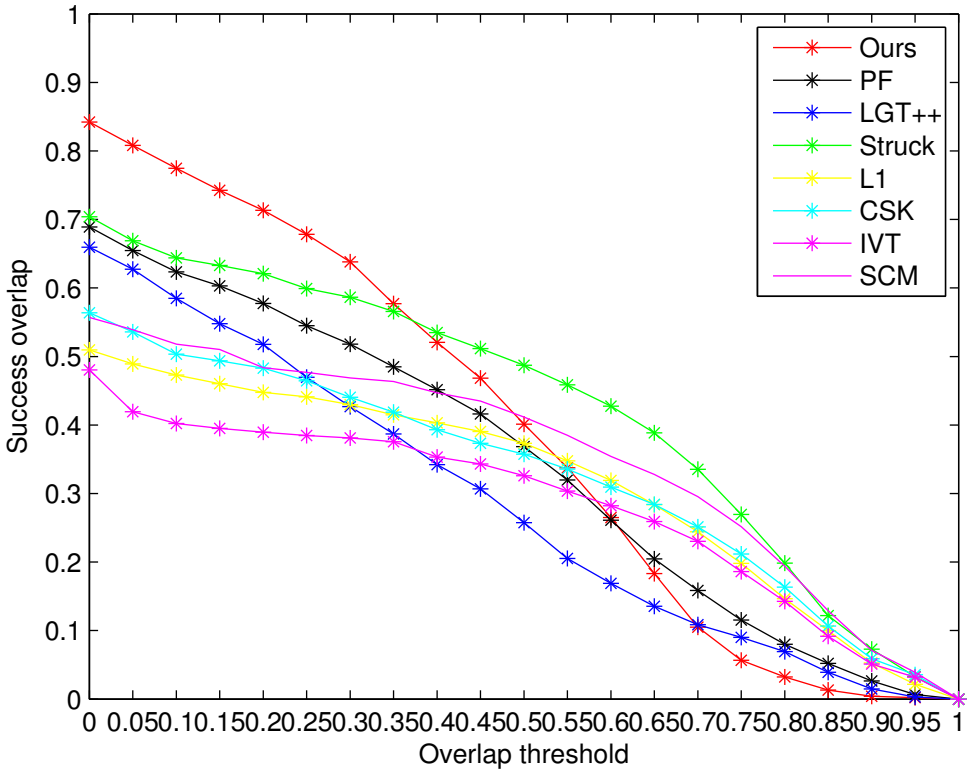

Figure 3.9: The success ratio versus overlap threshold curve in 22 sequences with occlusion.

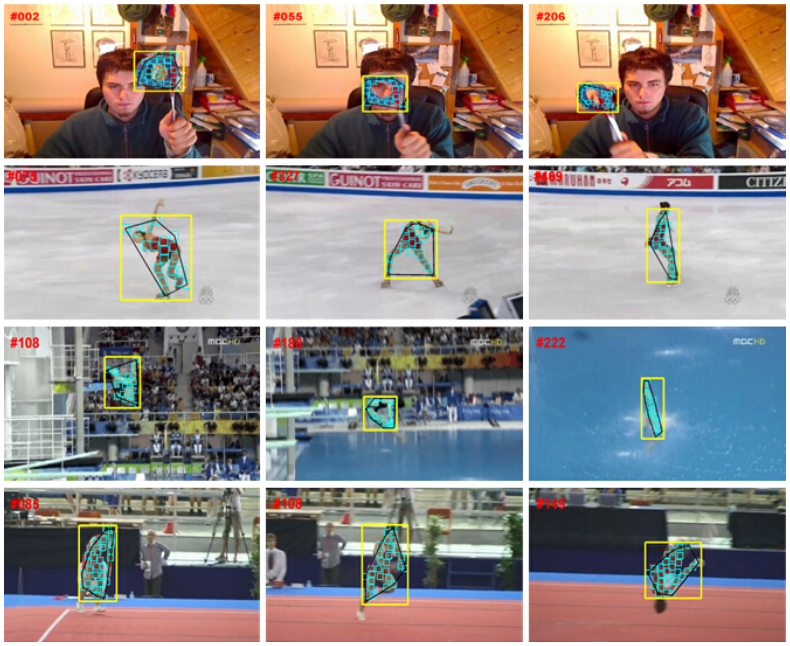

Figure 3.10: visualization of results on sequences: *Torus*, *Iceskating*, *Diving*, *Gymnastics*, showing extreme target deformations and significant clutter. Yellow bounding boxes: the estimated targets; (small) blue bounding boxes: matched local parts.

random target regions to be mid-layer patches. Secondly, we remove our proposed adaptive clustered decision trees (which adaptively combine different features according to their importance) and replace them with a standard homogeneous feature fusion method, as in [6] and much of the feature fusion literature, wherein the likelihoods of different features are simply multiplied. Thirdly, we remove our proposed adaptive model re-learning speed, and replace it with a fixed re-learning speed of 0.5 in equation 3.20. Figure 3.11 compares the performance of the complete proposed tracker, with performance of the tracker when each sub-component is removed and replaced by a conventional method from the literature.

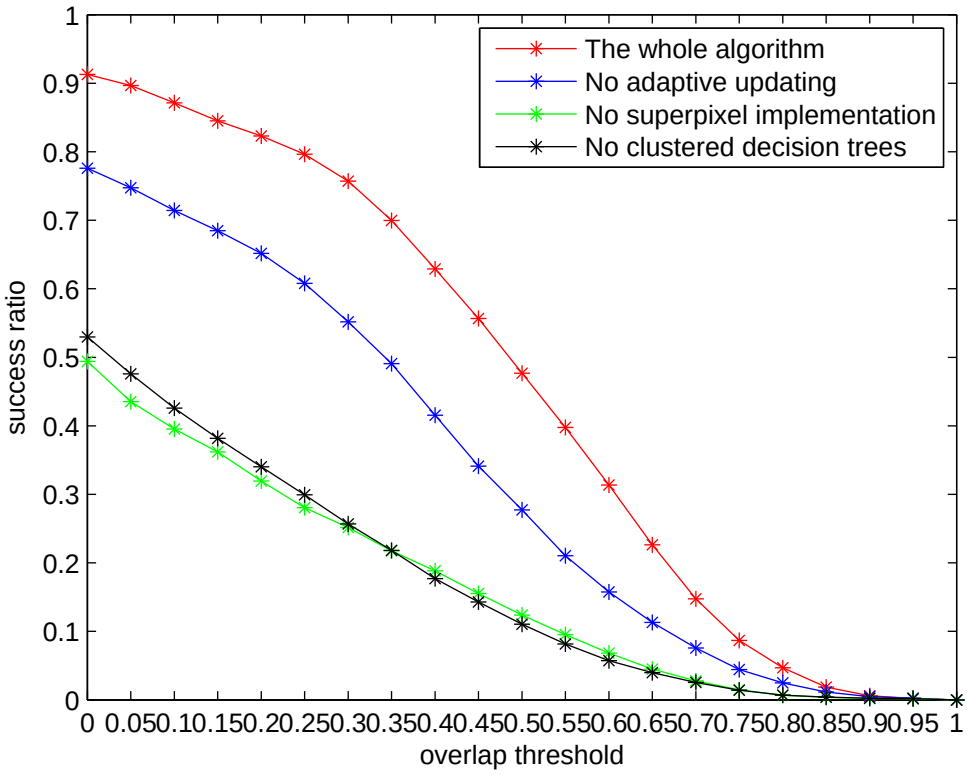

Figure 3.11: Success ratio versus overlap threshold curve, for 70 sequences. Comparison of proposed tracker versus modified tracker with each of three key components replaced by conventional methods from the literature.

Figure 3.11 suggests that the adaptive re-learning speed does have a significant impact on performance. However, even without the adaptive re-learning speed, the tracker is still able to deliver comparable performance to the state-of-the-art trackers analysed in figure 3.11. In contrast, removing either the super-pixels mid-layer parts representation or the clustered decision

tree matching method, causes a larger degradation in performance. It appears that the super-pixels representation and the clustered decision tree method contribute equally to the overall performance of the tracker, and that their contribution is more important than the effect (still significant) of the adaptive re-learning rate.

As discussed also in earlier sections, we suggest the following possible reasons for the effects of these techniques on overall performance. Firstly, because superpixel segmentation results in homogeneous image regions, it is more likely to lead to mid-layer patches which contain purely target pixels or purely background pixels. Target patches which erroneously contain background pixels will then be rapidly eliminated from the target model by other parts of the algorithm, leaving mid-layer patches which reliably adhere only to the target pixels, and this supports robust and stable tracking. Additionally, without super-pixel segmentation, the matching routine must exhaustively search all possible local candidate regions, as in [6], to achieve a robust match. One strong benefit of the clusterd decision tree, is that it adaptively weights in favour of whichever features are most discriminating for any particular patch, at any given frame. In contrast, conventional feature fusion (by multiplying feature likelihoods) enables a poorly performing feature (e.g. where the background is similar to the target in that feature modality) to damage the good performance which might be achieved by another feature that happens to be more discriminating in the current frame.

3.3 Summary

In this chapter, we have proposed a novel method for 2D RGB target tracking, comprising a multi-layer target model and an adaptive clustered decision tree method for both matching and relearning middle layer target parts at successive image frames. The adaptive decision tree: 1) is generated online, overcoming the overfitting of offline generated classifiers; 2) efficiently exploits only the minimum number of features needed for each target part at each frame; 3) adaptively weights in favour of the most discriminating features, responding dynamically to changing amounts of camouflage in different feature modalities. We test the tracker using two

different tracking benchmarks [2, 3], based on two different test methodologies, namely VOT and VTB. In VOT2013 tracking challenge, we achieve the best performance considering both robustness (no failure) and accuracy (0.59). In VOT2014 tracking challenge, we achieve the best robustness (one failure) and comparable accuracy (0.52). In VTB tracking benchmark, we achieve overall best success ratio up to the overlap with 0.4. To further evaluate the performance, we also show the trade-off curves for those test videos identified in the benchmark challenges as containing the categories of: significant target deformations; severe illumination changes; and occlusions. Our tracker significantly outperforms the other methods in tracking of highly deforming targets while still achieves competitive results in illumination change and occlusions scenarios [1]. We further investigate the extent to which each of the designing components, e.g. super-pixels based local parts, adaptive clustered decision trees and adaptive updating speed, contributes to the overall strong performance of the tracker. The experiment shows that the adaptive re-learning speed does have a significant impact on the performance. However, even without the adaptive re-learning speed, the tracker is still able to deliver comparable performance to the state-of-the-art trackers. In contrast, removing either the super-pixels mid-layer parts representation or the clustered decision tree matching method, causes a larger degradation in performance. It appears that the superpixels representation and the clustered decision tree method contribute similarly to the overall performance of the tracker, and that their contribution is more important than the effect (still significant) of the adaptive re-learning rate.

CHAPTER 4

RGB-D TRACKER

Due to the proliferation of affordable depth sensors, such as Microsoft Kinect, Asus Xtion and PrimeSense etc., depth information has attracted growing interest for target tracking. However, while RGB-based features can be comparatively invariant to target motion, depth information varies rapidly when the target moves towards or away from the camera. Therefore, this chapter presents a new RGB-D tracker which exploits spatio-temporal consistency of the features in RGB and depth images for adaptive fusion in the multi-layer target model. A physical constraint of the target with respect to other objects in the depth context is exploited, to adaptively estimate the range interval occupied by the target. This physical constraint encodes commonsense knowledge that the target parts cannot move behind background regions or in front of occluding objects.

Despite the availability of several high quality benchmark datasets for RGB tracker evaluation [3, 21, 22, 23], those available for RGB-D tracker evaluation are far more limited. Therefore, we have constructed a new RGB-D dataset which i) ensures maximum diversity with minimum size/cost through a rigorous bias analysis; ii) incorporates complete *per-frame* annotation for every frame in every sequence; iii) includes human annotation of three binary attributes, as well as statistical evaluation of seven more objective numerical attributes, for every frame; iv) includes videos from both moving and stationary cameras for every scene, which is necessary for disambiguating failures due to camera motion from other causes. All of the attributes are per-frame annotated to help: i) ensure unbiased dataset construction; ii) aid functional under-

standing and explanations; iii) assist researchers in choosing parameter values.

This chapter is organized as follows. The proposed RGB-D tracker is described in section 4.1 while the constructed dataset for evaluation is presented in section 4.2. Section 4.3 applies the proposed RGB-D tracking method to the newly proposed dataset, demonstrating state-of-the-art performance. The chapter is concluded in section 4.4.

4.1 Proposed method

Our proposed RGB-D tracker also represents the target with a multi-layer model: top layer (bounding box), middle layer (target parts) and bottom layer (pixels), derived from proposed the RGB tracker in chapter 3. The top layer propagates the overall target to candidate image regions (defined by candidate bounding boxes) and measures the confidence of each candidate by checking the temporal consistency of multiple features, including depth and colour. The middle layer matches smaller parts of the target to local regions within the bottom layer, and judges the saliency of each target part by comparing feature statistics of the target region and surrounding contextual region. Unlike earlier work [6, 7, 1] on coupled-layer RGB models, our method: 1) fuses multiple features within the top layer, and automatically triggers the use of the middle layer if ambiguity is detected in the top layer; 2) matches the local parts with cross-constrained likelihood, including local-part, global-target, and contextual information; 3) proposes a physical constraint of the target with respect to other objects in the depth context, to adaptively estimate the range interval occupied by the target.

4.1.1 Top layer tracking based on temporal consistency

The tracker first propagates a set of samples, using the top layer features, to find candidate target regions in both RGB and depth images. The candidate regions generated by each feature modality are adaptively fused to give an overall target estimation in the top layer.

A. Feature modalities

In RGB images, we extract colour attributes [145] defined as linguistic colour labels with eleven basic colour terms, namely black, white, brown, green, grey, orange, pink, purple, red, blue and yellow, which have been shown to perform well in object detection, tracking and recognition [146]. In depth images, we use depth HOG as the top depth feature. We apply those two features using kernelised correlation filters (KCF) for tracking [9], which are acknowledged for their high computational efficiency with the use of Fast Fourier transforms, explained below.

Correlation filters measure the similarity between the template and the image region to produce the correlation peaks for the target regions while low response for the background [80]. First, correlation filters are trained with the initialised image region (in our work, it is one bounding box). The algorithm extracts the features from the initialised region and often applies a cosine window to smooth the boundary effects. At the successive frames, correlation filters are conducted between the template and image candidate region using element-wise multiplications in Fourier domain instead of exhausted convolutions in Spatial domain [147]. This significantly improves the computation efficiency. Then, the correlation output is transformed back into the spatial domain as a spatial confidence map using the inverse Fast Fourier Transform. At last, the region with the maximum value indicates the new position of the target. More details could be found in [9].

In our work, we first apply the features extracted from RGB images and depth images with KCF [9], where the top target layer (bounding box) estimations at frame k are denoted as $\boldsymbol{R}^T_{rgb,k}$ and $\boldsymbol{R}^T_{d,k}$ respectively. Their corresponding matching scores are denoted as $\hat{p}(\boldsymbol{R}^T_{rgb,k})$ and $\hat{p}(\boldsymbol{R}^T_{d,k})$.

B. Temporal consistency based regression models

Based on the estimated target positions, $\boldsymbol{R}^T_{rgb,k}$ and $\boldsymbol{R}^T_{d,k}$, we distinguish two different tracking situations: "ambiguous" and "unambiguous" situations. A high overlap ratio between $\boldsymbol{R}^T_{rgb,k}$ and $\boldsymbol{R}^T_{d,k}$ indicates unambiguous tracking situations [148], and a low overlap ratio indicates an ambiguous situation.

In unambiguous situations, $\boldsymbol{R}^T_{rgb,k}$ and $\boldsymbol{R}^T_{d,k}$ represent similar target regions. They are used to estimate the final target region in the top layer, by assigning equal weights to $p(\boldsymbol{R}^T_{rgb,k})$ and

$p(\boldsymbol{R}^T_{d,k})$. These matching scores of the colour feature $\hat{p}(\boldsymbol{R}^T_{rgb,k})$ and depth feature $\hat{p}(\boldsymbol{R}^T_{d,k})$ are then used to train regression models, later used for measuring temporal consistency. These temporal consistency measures can then be used to robustly (adaptively) fuse $\boldsymbol{R}^T_{rgb,k}$ and $\boldsymbol{R}^T_{d,k}$ in ambiguous situations. More specifically, we first describe the procedure in the RGB image and denote its matching score as $\mathrm{f}_{rgb,k} = \hat{p}(\boldsymbol{R}^T_{rgb,k})$. The temporal history of a feature's matching scores in unambiguous frames is denoted as $\mathbf{F}_{rgb} = \{\mathrm{f}_{rgb,1}, .., \mathrm{f}_{rgb,k-1}\}$. It is used to train a linear regression model $\hat{\mathrm{f}}_{rgb,k} = \alpha k + \alpha_0$, to get $\alpha,\ \alpha_0$ online.

During *ambiguous* situations, training is switched off. The difference between the feature matching score in current frame $\mathrm{f}_{rgb,k}$ and a predicted score $\hat{\mathrm{f}}_{rgb,k}$ from the regression model, is used to measure temporal consistency. Feature weights are assigned by equation 4.1 where a feature with higher consistency score will be assigned a higher weight.

$$\hat{p}(\mathrm{f}_{rgb,k}|\mathbf{F}_{rgb}) = \exp(-|\mathrm{f}_{rgb,k} - \hat{\mathrm{f}}_{rgb,k}|) \tag{4.1}$$

In depth images, the weight $\hat{p}(\mathrm{f}_{d,k}|\mathbf{F}_d)$ of the used HOG feature is obtained in a similar way as the feature used in RGB images (equation 4.1). The final weights for fusion are computed by considering the temporal consistency of both RGB and depth features:

$$\begin{cases} p(\boldsymbol{R}^T_{rgb,k}) = \frac{\hat{p}(\mathrm{f}_{rgb,k}|\mathbf{F}_{rgb})}{\hat{p}(\mathrm{f}_{rgb,k}|\mathbf{F}_{rgb})+\hat{p}(\mathrm{f}_{d,k}|\mathbf{F}_d)}, \\ p(\boldsymbol{R}^T_{d,k}) = \frac{\hat{p}(\mathrm{f}_{d,k}|\mathbf{F}_d)}{\hat{p}(\mathrm{f}_{rgb,k}|\mathbf{F}_{rgb})+\hat{p}(\mathrm{f}_{d,k}|\mathbf{F}_d)}. \end{cases} \quad \text{if} \quad \boldsymbol{R}^T_{rgb,k} \cap \boldsymbol{R}^T_{d,k} > \lambda_\Delta \mathcal{O} \tag{4.2}$$

where $\mathcal{O}$ is the target size, $\lambda_\Delta = 0.9$ is a threshold to measure the ambiguity of $\boldsymbol{R}^T_{rgb,k}$ and $\boldsymbol{R}^T_{d,k}$. Here the value of λ_Δ is explicitly chosen to avoid the template drifting. In practice, when the estimation difference between tracker and ground-truth bounding box exceeds this value, the tracker's template will drift to the background after the model adaptation.

C. Estimation

The final estimate $\boldsymbol{R}^T_k$ for the top layer is obtained as a weighted linear combination of $\boldsymbol{R}^T_{rgb,k}$ and $\boldsymbol{R}^T_{d,k}$:

$$\boldsymbol{R}^T_k = p(\boldsymbol{R}^T_{rgb,k})\boldsymbol{R}^T_{rgb,k} + p(\boldsymbol{R}^T_{d,k})\boldsymbol{R}^T_{d,k} \tag{4.3}$$

Large differences between $\boldsymbol{R}^T_{rgb,k}$ and $\boldsymbol{R}^T_{d,k}$ indicate an "ambiguous" situation, where at least one feature is likely to be erroneous.

4.1.2 Parts-based tracking with spatial context

Once an "ambiguous" situation is detected, the algorithm will progress to the second (middle) layer for more accurate tracking. The top layer target region is enlarged, and then broken down into a set of local candidate regions (using [137]) for target parts matching.

A. Matching metrics

To match target parts to local image regions, we use three matching metrics in the previously proposed clustered decision trees, considering middle layer parts information, top layer target information and contextual information.

Middle layer colour based likelihood of target part q_t is the similarity between its colour histogram $\boldsymbol{M}^{c_m(q_t)}_{rgb,k}$ and that of local candidate image region p_s. To match target part q_t, local candidate regions are searched according to the criterion:

$$\begin{aligned} \hat{p}_s = arg\max\ \hat{L}^{c_m(q_t,p_s)}_{rgb,k}, \quad \hat{L}^{c_m(q_t,p_s)}_{rgb,k} = \mathcal{B}(\boldsymbol{M}^{c_m(q_t)}_{rgb,k}, \boldsymbol{M}^{c_m(p_s)}_{rgb,k}) \\ \text{s.t. } \hat{L}^{c_m(q_t,p_s)}_{rgb,k} > \lambda_{rgb}\hat{p}(\mathrm{f}_{rgb,k}|\mathbf{F}_{rgb}) \end{aligned} \tag{4.4}$$

where $\hat{L}^{c_m(q_t,p_s)}_{rgb,k}$ represents the colour likelihood and $\mathcal{B}(,)$ is the Bhattacharayya similarity coefficient [76]. The purpose of λ_{rgb} is to restrict the value $\hat{L}^{c_m(q_t,p_s)}_{rgb,k}$ to lie within a reasonable interval. In the RGB literature, matching likelihood $\hat{L}^{c_m(q_t,p_s)}_{rgb,k}$ is often used to indicate occlusion, if the *best* matching score is lower than a threshold. Previous methods [64, 1] set a fixed (minimum) likelihood threshold to determine occlusion, which may fail under variable tracking conditions, e.g. illumination changes. Therefore, we propose a temporal consistency-based feature weight, $\hat{p}(\mathrm{f}_{rgb,k}|\mathbf{F}_{rgb})$ in equation 4.1, to adaptively compute the constraint in equation 4.4.

Top layer colour based likelihood is obtained by applying top layer colour histogram to distinguish a pixel x in the estimated target region from its contextual pixels in RGB images [111], while suppressing distraction from background clutter. Let $\boldsymbol{M}^{c_t(\Omega)}_{rgb,k}(b^x_{rgb})$ be the b^x_{rgb} bin of the RGB histogram computed from (top layer) region $\boldsymbol{R}^{\Omega}_k \in \mathbf{I}_{rgb,k}$, where $\mathbf{I}_{rgb,k}$ represents the

RGB image and $\mathbf{I}_{rgb,k}(x) \mapsto b^x_{rgb,k}$. Given a rectangular target region $\boldsymbol{R}^T_k$ and contextual background $\boldsymbol{R}^C_k$ (illustrated in figure 3.2), we apply Bayes law to obtain the top layer colour based likelihood of pixel x as (similar definition as the bottom layer (pixel) model in equation 3.11):

$$l_k^{c_t(x)} \approx \frac{\hat{L}(b^x_{rgb}|x \in \boldsymbol{R}^T_k)\hat{L}(x \in \boldsymbol{R}^T_k)}{\sum\limits_{\boldsymbol{R}^\Omega_k = \boldsymbol{R}^T_k \cup \boldsymbol{R}^C_k} \hat{L}(b^x_{rgb}|x \in \boldsymbol{R}^\Omega_k)\hat{L}(x \in \boldsymbol{R}^\Omega_k)} \tag{4.5}$$

where $\hat{L}(b^x_{rgb}|x \in \boldsymbol{R}^T_k) \approx \boldsymbol{M}^{c_t(T)}_{rgb,k}(b^x_{rgb})$ while $\hat{L}(x \in \boldsymbol{R}^T_k) \approx \frac{|\boldsymbol{R}^F_k|}{|\boldsymbol{R}^T_k|+|\boldsymbol{R}^C_k|}$ where $|.|$ represents the region size. The notations with $\boldsymbol{R}^\Omega_k$ and $\boldsymbol{R}^C_k$ have similar definitions to those with symbol $\boldsymbol{R}^T_k$. Thus, the top layer colour based likelihood $\hat{L}^{c_t(p_s)}_{rgb,k}$ of the candidate part p_s considers all pixels inside local region $\boldsymbol{R}^{p_s}_k$, which is searched according to:

$$\hat{p}_s = arg\max \hat{L}^{c_t(p_s)}_{rgb,k}\ , \quad \hat{L}^{c_t(p_s)}_{rgb,k} = \sum_{x \in \boldsymbol{R}^{p_s}_k} l_k^{c_t(x)} \tag{4.6}$$

Top layer depth based likelihood is obtained from the depth histogram in a similar manner to the top layer colour based likelihood. Note that, due to target motion, target depth can change relatively fast (compared to colour), therefore the depth histogram used to compute likelihoods is continually re-estimated online. Here, we exploit physical constraints on the motions of the targets and other objects (e.g. a target cannot move deeper than background objects or come closer to the camera than an occluding object) in the depth context, to estimate the target histogram.

We first seek a depth interval in which the target might be located. [10, 20] computed this depth interval solely from the observed target depth histogram in frame $k-1$, which tightly constrains the target, figure 4.1, to lie between target foreground and background constraints ($D^f_{T,k-1}$ and $D^b_{T,k-1}$). Instead, we propose a physical constraint on the target pose wrt clutter objects, which allows us to relax those two constraints, enabling the proposed tracker to cope with large target depth variations.

We denote the observed target and contextual depth histograms at the $k-1$th frame as $\boldsymbol{M}^T_{d,k-1}$ and $\boldsymbol{M}^C_{d,k-1}$ respectively. Defining a narrow depth interval for the target may fail to

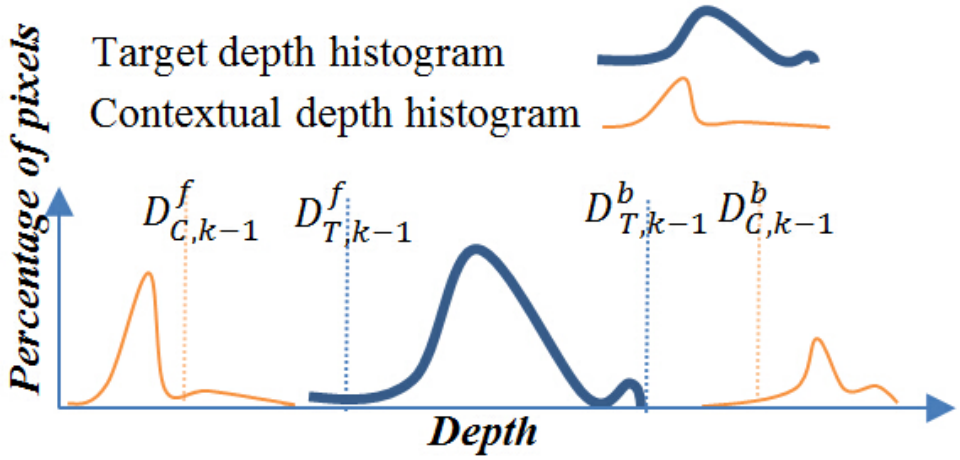

Figure 4.1: Depth histogram. $D^f_{T,k-1}$ and $D^b_{T,k-1}$ represent foreground and background constraints respectively, computed from the target depth histogram. Constraints $D^f_{C,k-1}$ and $D^b_{C,k-1}$ are computed from the contextual depth histogram.

handle target motion in the depth direction (equivalent to over-fitting wrt depth). Using too wide an interval could be regarded as under-fitting, and risks confusing target and background information causing matching difficulties. To overcome these issues, we initially compute the foreground and background depth constraints from depth histograms of both target and context separately, figure. 4.1. Between constraints $D^f_{T,k-1}$ and $D^b_{T,k-1}$ of the target's depth histogram, is a narrow interval which should contain a large proportion of target pixels. Between the context histogram constraints, $D^f_{C,k-1}$ and $D^b_{C,k-1}$, is a larger interval which we hope contains a small proportion of background pixels. We define information loss (proportion of pixels missing from a depth interval) as:

$$\begin{aligned} \mathcal{I}^T_{d,k-1} &= 1 - \sum_{b_d=D^f_{T,k-1}}^{D^b_{T,k-1}} \boldsymbol{M}^T_{d,k-1}(b_d) \\ \mathcal{I}^C_{d,k-1} &= 1 - \sum_{b_d=D^f_{C,k-1}}^{D^b_{C,k-1}} \boldsymbol{M}^C_{d,k-1}(b_d) \end{aligned} \tag{4.7}$$

where $\mathcal{I}^T_{d,k-1}$ and $\mathcal{I}^C_{d,k-1}$ denote information loss for depth intervals obtained from the target $\boldsymbol{M}^T_{d,k-1}(b_d)$ and contextual $\boldsymbol{M}^C_{d,k-1}(b_d)$ depth histograms respectively.

We seek an optimal depth interval which minimizes target information loss but maximizes

contextual information loss, computed by:

$$(\hat{D}^f_{T,k-1}, \hat{D}^b_{T,k-1}) = arg\min \begin{cases} \mathcal{I}^T_{d,k-1}, \\ D^{bf}_{T,k-1}. \end{cases} \quad \text{s.t.} \begin{cases} \mathcal{I}^T_{d,k-1} < \lambda_d, \\ D^{bf}_{T,k-1} > 0. \end{cases} \tag{4.8}$$

$$(\hat{D}^f_{C,k-1}, \hat{D}^b_{C,k-1}) = arg\max \begin{cases} \mathcal{I}^C_{d,k-1}, \\ D^{bf}_{C,k}. \end{cases} \quad \text{s.t.} \begin{cases} \mathcal{I}^C_{d,k-1} > 1 - \lambda_d, \\ D^{bf}_{C,k} > 0. \end{cases} \tag{4.9}$$

where$D^{bf}_{C,k} = D^b_{C,k} - D^f_{C,k}$ and $D^{bf}_{T,k-1} = D^b_{T,k-1} - D^f_{T,k-1}$.$\lambda_d$ is a threshold parameter to ensure that the target depth interval contains sufficient pixels. Equation 4.8 and equation 4.9 are a multi-objective optimization problem which we solve by combinatorial search, to yield an *adaptive depth interval*:

$$\begin{aligned} D^f_{A,k-1} &= \frac{D^f_{T,k-1} + \sum_{\phi\in\{b,f\}} \mathcal{H}(D^f_{T,k-1} - D^\phi_{C,k-1}) D^\phi_{C,k-1}}{1 + \sum_{\phi\in\{b,f\}} \mathcal{H}(D^f_{T,k-1} - D^\phi_{C,k-1})} \\ &\quad s.t. \quad D^f_{T,k-1} > D^f_{C,k-1} \\ D^b_{A,k-1} &= \frac{D^b_{T,k-1} + \sum_{\phi\in\{b,f\}} \mathcal{H}(D^b_{T,k-1} - D^\phi_{C,k-1}) D^\phi_{C,k-1}}{1 + \sum_{\phi\in\{b,f\}} \mathcal{H}(D^b_{T,k-1} - D^\phi_{C,k-1})} \\ &\quad s.t. \quad D^b_{T,k-1} < D^b_{C,k-1} \end{aligned} \tag{4.10}$$

where $\mathcal{H}$ is the Heaviside step function [149]. This ensures that, even if the context-based depth interval locates completely on one side of the target-based depth interval, both constraints can still be combined to generate an overall *adaptive depth interval*, defined by $D^b_{A,k-1}$ and $D^f_{A,k-1}$. This adaptive depth interval achieves robustness by encoding common-sense physical knowledge, that targets cannot recede beyond background, or approach closer to the camera than occluding objects. If these common-sense conditions are not met ($D^f_{T,k-1} \leqslant D^f_{C,k-1}$ or $D^b_{T,k-1} \geqslant D^b_{C,k-1}$) then an *ambiguity* situation in the depth modality is detected and the corresponding constraint becomes invalid.

In frame k, we regard any top layer pixels (those inside the 2D bounding box), which also lie within the *adaptive depth interval*, as belonging to the target. Mean depth $\bar{D}_k$ of those

target pixels is now used to *position-shift* the target depth histogram to the new expected target position:

$$\hat{\boldsymbol{M}}^{T}_{d,k-1}(b^{x}_{d}) = \boldsymbol{M}^{T}_{d,k-1}(b^{x}_{d} + \bar{D}_{k} - \bar{D}_{k-1}) \tag{4.11}$$

where $\boldsymbol{M}^{T}_{d,k-1}$ is the target depth histogram observed at frame $k-1$. $\mathbf{I}_{d,k}(x) \mapsto b^{x}_{d}$ is the b^{x}_{d} bin in the depth histogram, corresponding to pixel x in image $\mathbf{I}_{d,k}$. $\bar{D}_{k-1}$ is the mean depth of target pixels at frame $k-1$.

Note that a conventional target depth histogram cannot be used for tracking (e.g. in the same way as a colour histogram), because it is not a motion-invariant target feature. By position-shifting with respect to expected target motion, we have effectively created a *range-invariant* target depth histogram, $\hat{\boldsymbol{M}}^{T}_{d,k-1}$, which we can then use for tracking by generating likelihoods for target parts.

Next, the position-shifted target depth histogram in equation 4.11 and the contextual *depth* histogram $\boldsymbol{M}^{C}_{d,k-1}$ are used in a similar way as the target and context *colour* histograms are used in equation 4.5 and equation 4.6, to compute a top layer depth-based likelihood $\hat{L}^{d_t(p_s)}_{d,k}$ which can be used for matching the p_sth target-part. Using this depth likelihood we can now search for target parts matches according to the criterion:

$$\hat{p}_{s} = arg\max \sum_{x \in \boldsymbol{R}^{p_s}_{k}} \hat{L}^{d_t(p_s)}_{d,k} \quad \text{s.t.} \quad D^{f}_{A,k-1} < \frac{\sum_{x \in \boldsymbol{R}^{p_s}_{k}} \mathbf{I}_{d,k}(x)}{\sum_{x \in \boldsymbol{R}^{p_s}_{k}}} < D^{b}_{A,k-1} \tag{4.12}$$

B. Estimation

Our tracker applies the three matching metrics (middle layer colour, top layer colour and depth based likelihoods, equation 4.4, equation 4.6 and equation 4.12) to three levels of a clustered decision tree (see chapter 3), enabling matching of target parts q_t to local image regions p_s. The clustered decision tree initially attempts to match a part using a single feature (first level on the tree), and then progresses to additional features (deeper levels of the tree), if the initial feature (first level) yields several candidate regions p_s (a "cluster") which share similar

likelihoods to the best candidate p_s^{best}:

$$\mathbb{C}_{i,k}(p_s) = \begin{cases} 1, & \text{if } |\hat{L}_k^{i(p_s)} - \hat{L}_k^{i(p_s^{best})}| \leq \sigma_{i,k} \\ 0, & \text{otherwise} \end{cases} \tag{4.13}$$

where $\hat{L}_k^{i(p_s)}$ is ith feature likelihood of candidate local region p_s and $\sigma_{i,k}$ is the corresponding standard deviation. $\mathbb{C}_{i,k}(p_s)$ denotes whether or not region p_s is added to the cluster. For a particular target part, if the matching likelihood of any region p_s is sufficiently close to that of the best matching region p_s^{best}, then $\mathbb{C}_{i,k}(p_s)$ is set to 1, and region p_s is clustered with region p_s^{best}. $\mathbb{C}_{i,k}(p_s) = 1$ indicates that the current level of the decision tree (corresponding to use of a particular image feature) has failed to confidently match the target part to a unique image region. The tree then progresses to its next level (using an additional feature), to try and disambiguate candidate regions that remain clustered. When the target part finds a unique matching candidate region ($\sigma_{i,k}$ becomes zero), then the clustered decision tree algorithm terminates.

If no successful matching region can be found, then that target part is regarded as occluded. The final target position is computed from the distribution of all matched local parts. Each local part, associated with a clustered decision tree, also contributes a strong classifier for occlusion reasoning, by computing the ratio of matched local parts:

$$\mathbb{V}_k = N_{matched,k}/N_{all} \tag{4.14}$$

where $N_{matched,k}$ is the number of matched local parts and N_{all} is the number of all target parts. $\mathbb{V}_k$ can then be used as a "visibility" metric. When $\mathbb{V}_k$ becomes zero, then the target is fully occluded.

4.1.3 Model updating

It is crucial to update the target model to accommodate appearance changes, learn newly appearing information, and delete old information. The proposed tracker does this via two mech-

anisms: adapting the old middle and top layer target models, and adding new models of new middle layer target parts (derived from super-pixels), which is the same as explained in section 3.1.2.C.

Derived from proposed the RGB tracker in chapter 3, the proposed RGB-D tracker also represents the target with a multi-layer model: top layer (bounding box), middle layer (target parts) and bottom layer (pixels). Besides the novelties mentioned in chapter 3, the RGB-D tracker explicitly checks the temporal consistency of features in RGB images and depth images to predict the potential target region in the top layer, automatically triggering the use of the middle layer if ambiguity is detected. In addition, a physical constraint of the target (the target parts cannot move behind background regions or in front of occluding objects) with respect to other objects in the depth context is proposed to adaptively estimate the range interval occupied by the target, which improves the robustness of using depth information during tracking.

4.2 Dataset construction and bias analysis

Due to the limitations of currently available RGB-D benchmarks with respect to quality and quantity (as explained in section 2.4), we have created a new dataset to evaluate our proposed method. It consists of 36 video sequences with average 300 frames per sequence, shown in figure 4.2.

4.2.1 Dataset construction

4.2.1.A. Hardware set-up

There are a variety of options for choosing RGB-D sensors from the market (shown in figure 4.3), e.g. Microsoft Kinect and Asus Xtion etc. For depth images, Microsoft Kinect and Asus Xtion are similarly accurate for close ranges up to 3.5m with linear increased noise while noise increases quadratic in far ranges [151]. Microsoft Kinect offers various reliable hardware models and a remotely controllable motor is provided for device positioning. However, Kinect

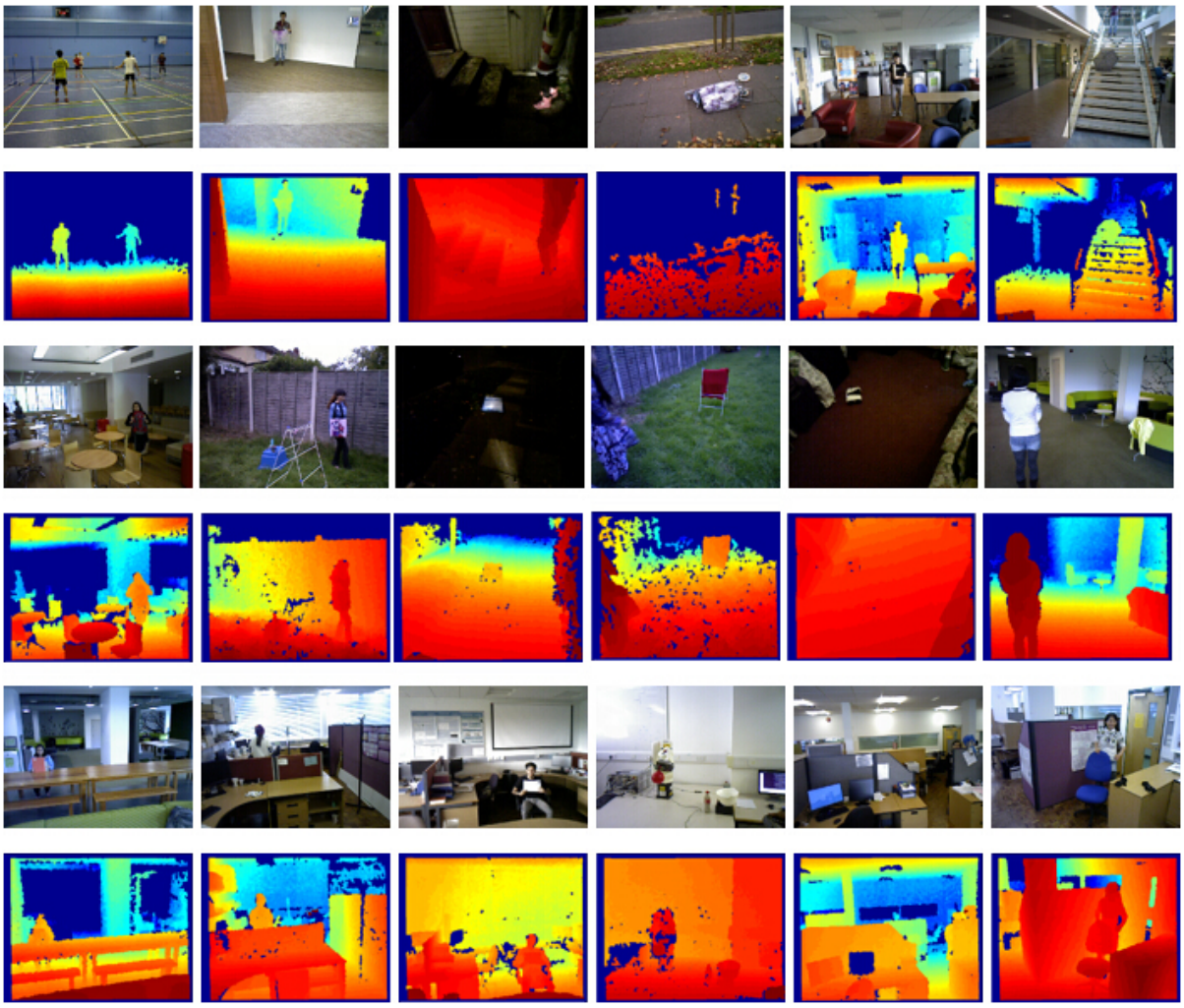

Figure 4.2: The constructed RGB-D tracking dataset. 36 video sequences (18 scenes) in total and each scene was captured with both a moving camera and stationary camera.

is relatively big (12” x 3” x 2.5”) and heavy (3.0 lb), while an additional AC-DC power supply is required. In contrast, Asus Xtion is more compact (7” x 2” x 1.5”) and lighter (0.5 lb). In addition, it does not need additional power supply and still provides better RGB image quality [150]. Considering some of the videos need to be captured outdoor, this makes Asus Xtion more convenient with high quality image data for our dataset construction.

Since motion of the RGB-D camera often causes significant depth variation, the moving camera brings more tracking challenges compared to the stationary camera. Therefore, in the experiment, two ASUS Xtion RGB-D sensors were used to collect data. For each scene, one sensor was fixed and the other sensor was moved continuously (by hand) to explicitly evaluate the performance of trackers under conditions of arbitrary camera motion. Note that the depth

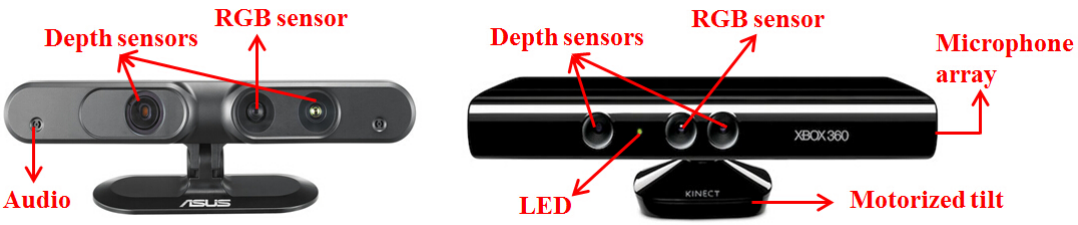

Figure 4.3: Different types of RGB-D cameras. The left one is Asus Xtion RGB-D sensor and the right one is Microsoft Kinect.

information is obtained by an embedded infrared projector in Xtion. Since the infrared will become saturated under conditions of direct sunlight, most sequences were recorded indoors with depth ranging from 0.5 to 8 meters. To ensure the diversity of the scenes and to investigate extreme tracking circumstances [152], some outdoor scenes were collected during the night. Note that even with the OpenNI2 synchronous function, RGB images and depth images are not always well synchronised [153], similar to problems in PTB [20] explained in section 2.4. Therefore, time stamps were used to drop asynchronous image pairs. Here, we calculated the difference of the time stamps between the RGB and depth image pairs. Once this time interval is higher than a threshold, we considered the RGB and depth images were unsynchronised. In the experiment, the threshold for time interval was set as 0.02s, where no apparent vision difference was visible to the human eye between the corresponding frames in the collected data.

4.2.1.B. Bounding box annotation

Human annotators were asked to manually select the minimum size of bounding box which fully contains the target object. As the dataset is designed for testing tracking algorithms (not for detection algorithms), the target objects were always fully or partially visible in every frame. In the case of partial occlusion, the bounding box covers both the visible and the (human-estimated) occluded region of the target object, as recommended by VOT [3].

Table 4.1: The descriptions of annotated attributes.

Attibutes	Description	Annotation method
IV	Illumination variation – RGB intensity change of all pixels inside the bounding box (mean value).	Computational statistics
DV	Depth variation – depth change of all pixels inside the bounding box (mean value).	
SV	Scale variation – scale change of the bounding box (relatively ratio).	
CDV	Color distribution variation – RGB distribution change of the bounding box.	
DDV	Depth distribution variation – depth distribution change of the bounding box.	
SDC	Surrounding depth clutter – depth similarity between the target and ring-shaped contextual region (mean value).	
SCC	Surrounding color clutter – RGB intensity similarity between the target and ring-shaped contextual region (mean value).	
BCC	Background color camouflages – if the object in the background shares the similar color as the target (binary value).	Human intuition
BSC	Background shape camouflages – if the object in the background shares the similar shape as the target (binary value).	
PO	Partial occlusion – if the target is partially occluded (binary value).	

4.2.1.C. Attributes annotation

"Attributes" define various challenging factors of tracking, and can provide a qualitative or functional interpretation of the characteristics of each sequence. To thoroughly analyse the dataset bias, we provide two categories of attributes annotations for each frame: binary and statistical, shown in table 4.1. The former are decided by human annotators based on intuition and include background colour camouflage (BCC), background shape camouflages (BSC) and partial occlusion (PO). The latter are calculated from the statistical properties of features within the annotated bounding boxes, including illumination variation (IV), depth variation (DV), scale variation (SV), colour distribution variation (CDV), depth distribution variation (DDV), surrounding depth clutter (SDC), and surrounding colour clutter (SCC).

4.2.2 Bias analysis

4.2.2.A. Tracking performance limits

To evaluate bias in the dataset [131], we first compute upper-limits and lower-limits that a tracker could achieve in each sequence, represented by overlap ratio between the bounding boxes of the tracker and the ground truth.

Upper-limits (green column heights in figure 4.4) represent ideal tracking performance (perfect match to ground truth). Since some tracking methods are not designed with scale adaptation, another upper-limit (red column heights in figure 4.4) is computed for a tracker which matches the true target centroid position but with the initialized (first-frame) target scale.

Lower-limits (blue columns in figure 4.4) represent the performance that can be achieved by a "dumb" tracker, defined as the best performing out of setting the bounding box: i) permanently in the image centre; ii) fixed at the initialisation location; iii) positioned randomly.

Sequences with high lower-limit should be avoided to avoid dataset bias [131], since arbitrary tracking algorithms tested on those sequences could achieve similarly good performance. Trackers without scale adaptation are best evaluated on sequences with high red column heights in figure 4.4, since those sequences have larger test range and the evaluation focuses on the center error not the scale adaptation. Also, for those trackers without scale adaptation, comparing their results to this upper-limit (red column height) is more objective and informative (compared to green column height). This is because no matter how accurate a tracker's position is, its tracking performance cannot exceed red column height (upper-limit). In contrast, trackers with scale adaptation are better tested on sequences with low red column heights, so that the scale adaptation of a tracker could significantly affect the tracking performance.

4.2.2.B. Attributes statistic

We first calculate the number of frames for each **binary attribute**, including background colour camouflage (BCC), background shape camouflages (BSC) and partial occlusion (PO), shown in figure 4.5. To avoid bias, the dataset should have a wide spread of attributes, which is shown as

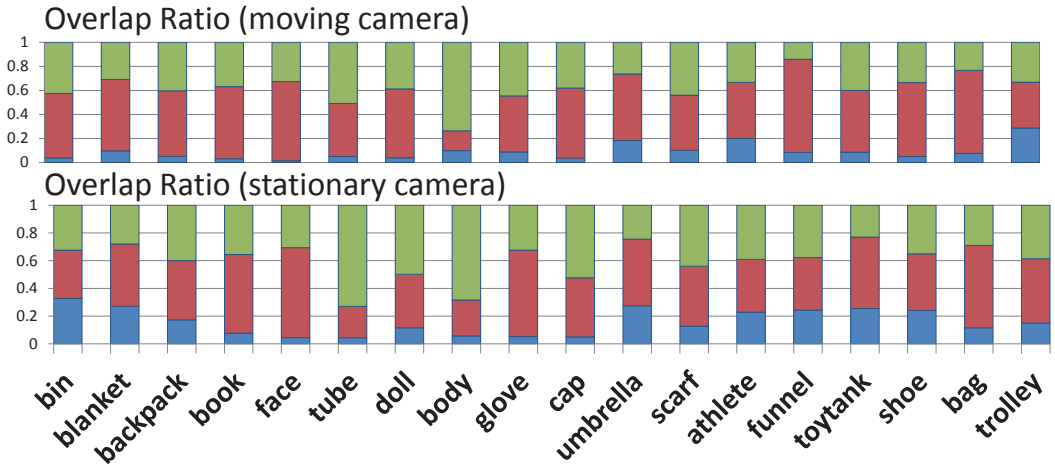

Figure 4.4: Limits of tracking performance in each sequence. Blue, green and red column heights correspond to lower limits and upper limits with/without scale adaptation, respectively.

different colour sections in bars. It can be seen that over half of the scenes have the attributes of BCC and BSC, which often last a long time. In contrast, partial occlusion happens more often across different scenes but just lasts a few frames. This supports our argument that the sequences should be per-frame annotated since the annotated attributes do not last throughout the entire sequence. It is preferable to evaluate each tracker with respect to attributes annotated on a per-frame basis, for a more accurate and meaningful analysis. Then, we use the number of attributes per frame to evaluate the tracking difficulties. To appreciate how challenging our dataset is, as shown in figure 4.6, the majority of image frames in the dataset contain either one or two challenging attributes, and a few frames are extremely challenging (containing all binary attributes).

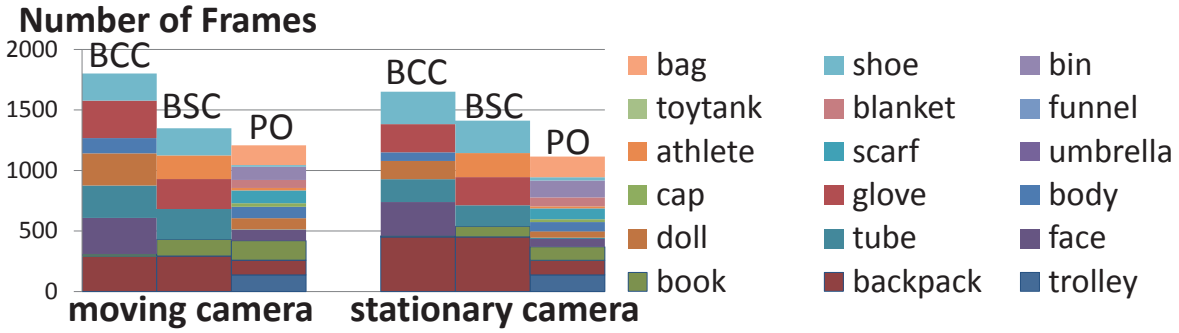

Figure 4.5: Number of frames for each binary attribute, including background colour camouflage (BCC), background shape camouflages (BSC) and partial occlusion (PO). Different colour sections represent different sequences.

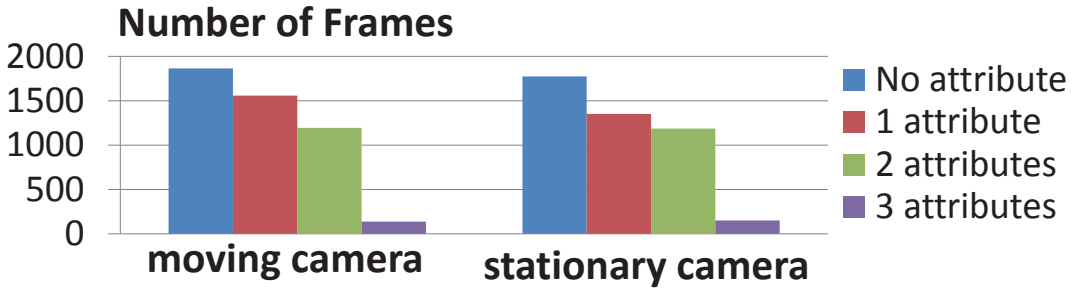

Figure 4.6: Proportions of dataset frames which contain various numbers of challenging (binary) attributes, including BCC, BSC, PO.

Statistical attributes, such as IV, DV, SV, CDV, DDV, SCC and SDC are computed from all frames in all sequences. For IV and DV, we compute the mean value of all pixels inside the bounding box in terms of intensity and depth. The differences of those values between successive frames are used to represent those two statistical attributes. For CDV and SCC, we use (intensity or depth) value of all the pixels inside the bounding box to compute the distribution which is compared (by using Bhattacharyya coefficient) to the distribution in the next frame to represent CDV or DDV. SCC and SDC compute the (intensity or depth) distributions of the pixels inside the bounding box and ring-shaped regions (figure 3.2) at the same frame, respectively. Then, we compare the distributions between bounding box and ring-shaped regions with Bhattacharyya coefficient, which is used to present SCC/SDC. SV used the scale of the bounding box between successive frames. The changes of the target scale are normalized by the absolute value of the bounding box scale.

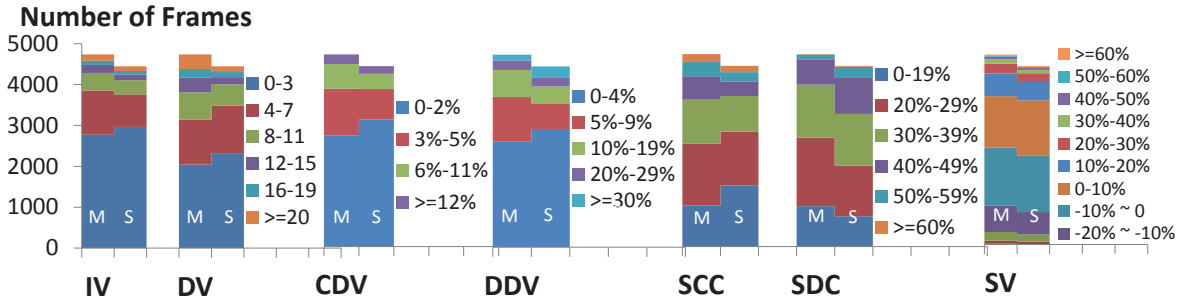

Figure 4.7: The statistical attributes of constructed dataset. Colour sections denote different ranges of severity of each attribute. IV, DV are measured by absolute value change (depth normalized 0 to 255). CDV, DDV, SBC, SCC are based on histogram while SV measures scale change ratio, defined in table 4.1.

Figure 4.7 suggests that there is a good spread of different attributes, and each has a good spread of severities, over the frames of the dataset. These statistical attributes are useful to: 1) guide unbiased dataset construction; 2) provide functional explanations of tracking failures; 3) help researchers choose optimal algorithm parameters.

4.3 Experiments

We have tested our RGB-D tracker on our new dataset using two (currently competing) state-of-the-art evaluation protocols, VTB [23] (runs each tracker throughout each sequence without reinitialisation following failures), and VOT [3] (re-initialises trackers whenever a failure is detected). Our tracker is compared against the top three [5] ranked RGB-D trackers: PT tracker [20], SD-KCF tracker [10], and OAPF tracker [148]. Test results are presented for i) the entire dataset; ii) stationary vs. moving camera; iii) frames corresponding to particular attributes.

4.3.1 VTB evaluation protocol

For VTB evaluation [21], there is no re-initialisation after tracking failures. [23] suggests area under curve (AUC) of the overlap ratio curve, or centre-error distance curve for evaluation. The centre-error measure (distance between tracker and groundtruth centroids) includes all frames, even after total tracking failure, which does not reflect the true performance of the tracker [154]. Moreover, centre-error is sensitive to subjective human bounding box annotation which can be significantly variable. Therefore, we compare each tracker in terms of overlap ratio curve. Table 4.2 shows the AUC performance of each tracker. Our proposed tracker demonstrates best overall performance, as well as best performance in all attribute sub-sets. PT [20] ranks second overall, while SD-KCF [10] outperforms PT and OAPF in the subsets of moving cameras, IV, DV, SV, DDV and SDC. OAPF ranks at bottom according to the VTB protocol. The DV severity at tracking failure frames is 6.4 for the proposed tracker and OAPF, and 5.9 and 5.7 for

Table 4.2: AUC of overlap curve in VTB evaluation protocol. (red: best performance; blue: second best performance)

	Ours	PT [20]	SD-KCG [10]	OAPF [11]
Overall	8.34	7.45	7.17	5.24
Stationary	9.57	8.54	7.52	6.00
Moving	7.18	6.43	6.85	4.54
IV	5.78	4.15	5.50	3.19
DV	7.53	6.71	6.97	4.45
SV	5.05	2.95	3.20	3.09
CDV	5.11	0.43	1.43	3.22
DDV	7.64	3.54	4.17	3.70
SDC	7.99	6.83	7.79	4.99
SCC	9.51	8.25	8.09	6.12
BCC	6.67	5.74	4.82	3.79
BSC	7.13	5.81	4.99	4.81
PO	7.74	6.16	6.03	5.81

DS-KCF and PT respectively. The tracking performance of the trackers in some severe scenes is visualized in figure 4.8.

4.3.2 VOT evaluation protocol

VOT evaluation [3] is based on two independent metrics: *accuracy* (overlap ratio between tracker and ground truth bounding boxes); and *robustness*, measured according to the frequency of tracking failure (when overlap ratio becomes zero). Whenever failures occur, the tracker is accurately reinitialised to continue tracking.

Failure rates and accuracy scores are shown in Tab. 4.3. Our proposed tracker ranks first in robustness, with competitive accuracy, while SD-KCF [10] ranks second in robustness (with seemingly better accuracy). Note that accuracy scores can be misleading. Since ground truth is used to re-initialise trackers (with perfect accuracy) following every tracking failure, less robust trackers (many failures) are likely to exhibit artificially higher accuracy scores. Therefore, the accuracy metric *is only meaningful when comparing two trackers which have the same robustness score*. Our method's robustness score significantly exceeds all others, while having a somewhat smaller accuracy score than competitor DS-KCF [10]. Meanwhile, PT [20] fails significantly and the accuracy of OAPF [11] ranks bottom. When failure happens, the average

Figure 4.8: Visualization of results on sequences: *Athlete*, *Backpack*, *Face*, *Trolley*, showing extreme target deformations and depth variation.

DV severity for our proposed tracker is 9 (normalised depth ranges from 1-255) while OAPF closely follows with 8.9. PT and DS-KCF are approximately 8.5 and 7, respectively.

4.3.3 Overall tracker evaluation: VTB *vs* VOT

The proposed tracker significantly and consistently performs best, on both VTB and VOT evaluation protocols. However, note that the performance of PT [20] changes dramatically between protocols (worst in VOT but second best in VTB). This is because PT often loses the target and then successfully re-captures it later, while VOT anyway re-initialises all trackers immediately following failures. Therefore, we suggest VOT is less suitable than VTB for testing trackers which are good at automatically recovering from failures. However, in VTB, success ratio can

Table 4.3: VOT protocol evaluation results. (red: best performance; blue: second best performance)

	Ours		PT [20]		SD-KCF [10]		OAPF [11]	
	Fail.	Acc.	Fail.	Acc.	Fail.	Acc.	Fail.	Acc.
Overall	2.11	0.52	10.61	0.64	3.36	0.56	5.78	0.35
Stationary	1.56	0.54	8.61	0.63	3.00	0.56	3.72	0.37
Moving	2.67	0.50	12.61	0.64	3.72	0.57	7.83	0.33
IV	0.11	0.41	0.33	0.53	0.08	0.50	0.19	0.25
DV	0.5	0.49	3.14	0.65	0.64	0.56	1.42	0.32
SV	0.19	0.41	1.42	0.55	0.28	0.48	0.50	0.25
CDV	0	0.46	0.19	0.60	0.03	0.54	0.11	0.22
DDV	0.25	0.47	1.03	0.65	0.31	0.58	0.61	0.27
SDC	0.97	0.52	5.78	0.62	1.39	0.55	2.75	0.35
SCC	0.22	0.52	2.33	0.62	0.92	0.52	1.36	0.36
BCC	1.17	0.49	4.31	0.64	1.56	0.53	2.61	0.32
BSC	0.67	0.50	3.83	0.63	1.47	0.51	1.86	0.35
PO	0.5	0.46	3.61	0.57	1.11	0.54	1.47	0.33

be dramatically affected by the sequential positions of failures. If a tracker fails early in a sequence, it will show a low success ratio, but if it fails late in a sequence, it may show high success ratio. Therefore, we evaluate our tracker with both VOT and VTB methodologies, and it shows superior performance in both.

4.4 Summary

This chapter has presented a novel method for RGB-D target tracking, and also presented a new benchmark dataset. The proposed tracker extends the original RGB multi-layered target model, which robustly combines both RGB and depth information. In the top layer, the algorithm exploits temporal consistency to adaptively fuse features for candidate region searching. The global candidate region is then split into a set of local candidate regions for robust matching to local target parts. Parts matching is robustified by considering both top layer feature statistics and contextual information. A physical constraint is extracted from the depth context to accurately estimate an adaptive target depth interval.

To evaluate our tracker, we developed a new RGB-D benchmark dataset with *per-frame*

annotated attributes. We compute upper-limits and lower-limits that a tracker could achieve in each sequence to analyse the bias in dataset. We provide two categories of attributes annotations for each frame: binary and statistical. The former are decided by human annotators based on intuition, and the latter are calculated from the statistical properties of features within the annotated bounding boxes. We calculate the distribution of different attributes over the frames of the dataset, which suggests that there is a good spread of different attributes, and each has a good spread of severities. These attributes are useful to: 1) guide unbiased dataset construction; 2) provide functional explanations of tracking failures; 3) help researchers choose optimal algorithm parameters.

Our tracker is evaluated using two different state-of-the-art evaluation protocols [3, 23]. The proposed tracker significantly and consistently performs best, on both VTB and VOT evaluation protocols. For VTB protocol, our proposed tracker demonstrates best overall performance, as well as best performance in all attribute sub-sets. The depth variation severity at tracking failure frames is 6.4 for the proposed tracker, which is equally good as OAPF. For VOT protocol, the proposed tracker ranks first in robustness, with competitive accuracy. When a failure happens, the average depth variation severity for our proposed tracker is 9 (normalised depth ranges from 1-255), which accommodates largest depth range compared to the other trackers.

CHAPTER 5

TRACKING BY EXPLOITING SCENE DISTRACTORS

Previous proposed work built the tracker by focusing on modelling and reasoning about the tracked object itself. This chapter proposes that, to achieve robust tracking under severe tracking conditions, the tracking algorithm needs to go beyond the target information and "interact" with the context in a sophisticated way, by locating and modeling multiple regions in each image which share similar appearance with the target. We call such image regions *distractors*. Therefore, in this chapter, we present a novel method for single target tracking in RGB images under conditions of extreme clutter and camouflage, including frequent occlusions by objects with similar appearance to the target. We first propose a multi-level clustering method for online detection and learning of multiple distractors. Section 5.1 explains our proposed method. To distinguish the target from these distractors, we use global dynamic constraints (derived from target and distractor information) to optimize the estimated target trajectory. Section 5.2 presents performance evaluations of our method, and comparisons against other state-of-the-art algorithms. To evaluate our tracker, we have augmented publicly available benchmark videos, by proposing a new set of clutter sub-attributes, and annotating these sub-attributes for all frames in all sequences. Using this dataset, our proposed method outperforms the best ranked state-of-the-art single target trackers on highly cluttered scenes. Furthermore, for tracking one out of many identical entities, our method outperforms state-of-the-art multi-target trackers, when they are given additional prior knowledge (initialised with all distractors as separate targets in the first

frame). Section 5.3 summarises this work and provides concluding remarks.

5.1 Proposed method

The proposed method consists of two steps. The first step uses coarse-to-fine multi-level clustering to find candidate image regions for target and/or distractors (the regions that share similar appearance with the target). The second step aims to distinguish the target from the distractors by using a global dynamic constraint based on the motion history of both tracker and distractors.

5.1.1 Coarse-to-fine multi-level clustering

Our proposed tracking algorithm first propagates a set of samples drawn from the region around the target position estimated at the preceding frame. We then propose a multi-level clustering method to find regions that are similar to the target in the new frame. Multiple feature modalities and spatial information are used, level by level, in a coarse-to-fine sampling manner to incrementally achieve better results, shown in figure. 5.1. This approach resolves the contradiction between robustness and tracking speed, by first performing sparse sampling to find initial candidates, and later applying dense sampling to a small subset of image regions (identified by the sparse sampling) where needed.

The algorithm begins by propagating only a sparse set of samples, and colour features are initially used to compute matching scores for each sample. First, clustering is carried out according to colour matching scores to classify the samples into *foreground* and *background* sets (level 1 clustering), defined as those with high and low matching scores respectively. Next, the spatial distribution of level 1 foreground samples is used to sub-cluster neighbouring samples (level 2 sub-clustering). For each level 2 cluster, we then apply a dense sampling, using an additional feature modality (HOG) for robust estimation (level 3 cluster subdividing). Note that we use the term *foreground* in a special sense, to denote *both target and distractors*. Everything else is called *background*.

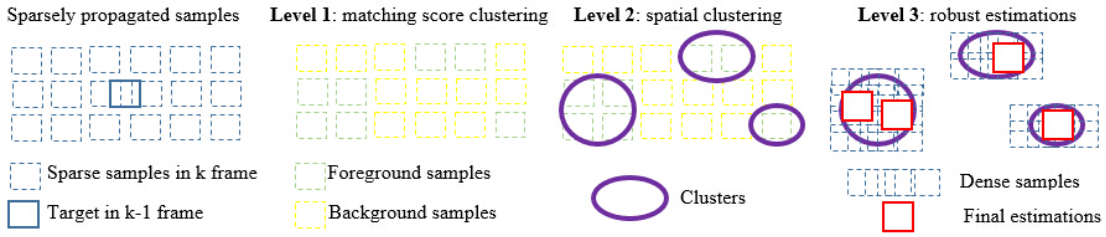

Figure 5.1: Coarse-to-fine multi-level clustering is used to find image regions containing the target or distractors. The algorithm: i) sparsely propagates samples around the target position from the previous frame; ii) clusters the propagated samples into two groups (foreground/background samples) according to their associated matching scores; iii) clusters the foreground samples according to their spatial distribution; iv) densely samples each clustered foreground region to perform robust estimations.

A. Level 1 clustering

The algorithm samples a sparse set of N_R locations surrounding the target location in a uniform way. The positions of the sample centres at the kth frame are denoted by $\{\hat{\mathbf{X}}_k(j)\}_{j=1,\ldots,N_R}$. As colour histograms are acknowledged for their simplicity, computational efficiency, invariance to scale and resolution change [64], we first extract a colour histogram from each sample and compare it to the target appearance model to get matching scores $\{\hat{L}_{C,k}(j)\}_{j=1,\ldots,N_p}$, where C indicates the colour feature. Within the information of the sample distributions and their associated matching scores, we use $\mathbf{f}_k^j = \{\hat{\mathbf{X}}_k(j), \hat{L}_{C,k}(j)\}$ as the feature vector and feed them into a Gaussian Mixture Model in order to cluster the samples into two groups: foreground samples and background samples, by equation 5.1.

$$p(\mathbf{f}_k^j;\theta) = \sum_{\mathbb{C}=1}^{2} \alpha_{\mathbb{C}} \mathcal{N}(\mathbf{f}_k^j; \mu_{\mathbb{C}}, \sum\nolimits_{\mathbb{C}}) \tag{5.1}$$

where $\alpha_{\mathbb{C}}$ is the weight of the cluster $\mathbb{C}$, $0 < \alpha_{\mathbb{C}} < 1$ for all components, and $\sum_{\mathbb{C}=1}^{2} \alpha_{\mathbb{C}} = 1$, where μ and $\sum$ are the mean and variance of the corresponding cluster. The parameter list:

$$\theta = \{\alpha_1, \mu_1, \sum\nolimits_1, \alpha_2, \mu_2, \sum\nolimits_2\} \tag{5.2}$$

defines a Gaussian mixture model, which is estimated by maximising the likelihood [155]. The mean matching score of samples in each cluster is denoted by $\bar{L}_{C,k}^{\mathbb{C}}$. Then, all samples in the

cluster with greatest mean score are regarded as foreground samples, denoted as equation 5.3.

$$\mathbb{R}_f(\hat{j}) = \begin{cases} 1, & \text{if } \hat{j} = \arg\max \bar{L}^{\mathbb{C}}_{C,k}(j) \\ 0, & \text{otherwise.} \end{cases} \tag{5.3}$$

where $\mathbb{R}_f$ denotes whether sample $\hat{j}$ is regarded as foreground. The selected foreground samples $\hat{\boldsymbol{R}}_k(\hat{j})$ will next be used for level 2 sub-clustering using additional features, while all samples in the cluster with lower mean score are regarded as background and are discarded.

B. Level 2 sub-clustering

In a highly cluttered environment, there may be many false positives among those samples labeled as foreground, caused by distractors (non-target image regions with target-like appearance). To distinguish individual objects in the scene (target and distractors) we therefore sub-cluster the samples within all level 1 clusters according to their spatial distribution:

$$\mathbb{C}_{sub}(i,j) = \begin{cases} 1, & \text{if } \mathbb{N}(i,j) = 1, \;\; i,j \in \mathbb{R}_f \\ 0, & \text{otherwise.} \end{cases} \tag{5.4}$$

where $\mathbb{N}(i,j)$ denotes whether sample i and j are neighbours. $\mathbb{C}_{sub}(i,j) = 1$ labels samples i and j as belonging to the same sub-cluster. $\mathbb{R}_f$ represents the foreground sample cluster (level 1 cluster). Noticeably, the performance of this spatial distribution-based clustering method depends on the spatial density of propagated samples. If the samples are sparsely distributed, it is likely that a level 2 sub-cluster may contain more than one object (a similar problem was identified in [121, 122]). Figure 5.2 illustrates the results after level 1 and level 2 clustering, using a frame from the $Juggling$ sequence. Even if there is a gap between two adjacent objects (Figure 5.2. A), it can be difficult to distinguish them using a sparse sampling density, Figure 5.2. B. Therefore, we next proceed to another level (level 3 cluster subdividing), where the foreground regions identified by levels 1 and 2 are more densely sampled and an additional appearance feature is added to achieve finer scale disambiguation.

C. Level 3 cluster subdividing

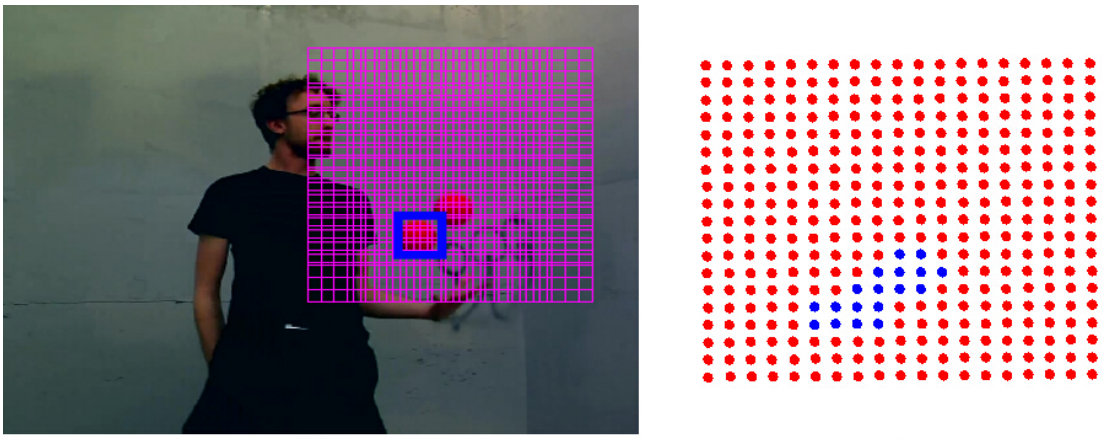

Figure 5.2: Failure mode of levels 1 and 2 clustering. In this image, the true target (red ball) is very close to a distractor (another identical red ball), which can lead to failure. (A) red grid denotes the sparsely distributed samples. Blue rectangles are the estimated foreground regions; (B) red dots denote background samples while blue dots denote foreground samples which have been erroneously merged into a single foreground cluster. Clearly levels 1 and 2 clustering can fail to disambiguate two adjacent objects.

In this level, the algorithm will estimate the regions of tracker or distractors from each sub cluster. In the initial stage, we sparsely sample the image to achieve computational efficiency in a large searching scope. After we obtain the set of foreground samples from levels 1 and 2 clustering, we densely sample the region inside each level 2 sub-cluster to further improve the localisation of target and distractor regions. Each level 2 sub-cluster is obtained using colour features for matching, and all level 2 foreground samples therefore already have a high colour matching score. Therefore an additional feature modality is needed to achieve further disambiguation of target and distractor regions. At level 3 the algorithm therefore applies HOG features to compute the matching scores of the new samples, using a kernel correlation filter [144].

Within each densely re-sampled level 2 sub-cluster, the most straightforward way to identify the object region is to search for the sample with the highest HOG feature matching score. However, as shown in figure 5.2, sometimes a coarse level 2 cluster may contain more than one object. If the target undergoes deformation, then a distractor within the same cluster often triggers a high matching score. [64, 112] tried to detect the target by applying the expectation operator over the distributed samples with associated weights (i.e. taking the likelihood-weighted mean of all samples). However, the expectation estimation might be highly erroneous when

multiple similar objects are present in the scene [156]. For example, taking the mean location of two similar objects will give an estimated location which lies on a background region, midway between both samples, shown in figure 5.3. To overcome this problem, we observe that the spatial ambiguity between the sample with the highest matching score (the *mode*) and the location of the *mean* sample (derived from the expectation operator) can indicate potential distractions within a cluster, and enable robust estimation.

Figure 5.3: Tracking failure. Taking the mean location of two similar objects will give an estimated location which lies on a background region.

Within the dense level 3 samples, the initial estimate of the object inside each cluster is taken to be the sample with highest HOG matching score (i.e. the mode sample), denoted by $\hat{\boldsymbol{R}}_k^{\mathbb{C}_{sub}}(j_h)$. We also use the expectation operator over all samples in the cluster to compute the mean sample:

$$\bar{\boldsymbol{R}}_k^{\mathbb{C}_{sub}} = \sum_{j=1}^{N_{\mathbb{C}}} \hat{L}_{H,k}^{\mathbb{C}_{sub}}(j) \hat{\boldsymbol{R}}_k^{\mathbb{C}_{sub}}(j) \tag{5.5}$$

where $\hat{L}_{H,k}^{\mathbb{C}_{sub}}(j)$ is the associated HOG feature matching score of the dense sample j inside level 2 sub-cluster $\mathbb{C}_{sub}$ and $N_{\mathbb{C}}$ is the number of samples inside each cluster. If the overlap between $\hat{\boldsymbol{R}}_k^{\mathbb{C}_{sub}}(j_h)$ and $\bar{\boldsymbol{R}}_k^{\mathbb{C}_{sub}}$ is small, it suggests there is another distractor inside the cluster, which is on the opposite side of $\hat{\boldsymbol{R}}_k^{\mathbb{C}_{sub}}(j_h)$ compared to $\bar{\boldsymbol{R}}_k^{\mathbb{C}_{sub}}$, see figure 5.4. If we denote foreground samples in the other half of the cluster as $\mathbb{R}_{f,\mathbb{C}_{sub}/2}$, then a second object's location is estimated by:

$$\hat{j} = \arg\max \ \hat{L}_{H,k}^{\mathbb{C}_{sub}}(j)$$
$$\textbf{s.t.} \ \ i \in \mathbb{R}_{f,\mathbb{C}_{sub}/2}, \ \ \hat{\boldsymbol{R}}_k^{\mathbb{C}_{sub}}(j_h) \cap \bar{\boldsymbol{R}}_k^{\mathbb{C}_{sub}} < \zeta \tag{5.6}$$

where ζ is the overlap threshold. This method will iteratively estimate the potential distractors inside each cluster until $\hat{\boldsymbol{R}}_k^{\mathbb{C}_{sub}}(j_h)$ and $\bar{\boldsymbol{R}}_k^{\mathbb{C}_{sub}}$ have significant overlap. Here, we use the threshold 50% to judge whether the overlap is significant.

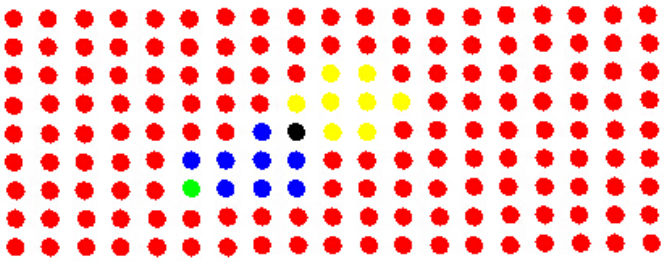

Figure 5.4: Sub-image of figure 5.2.B with level 3 clustering. Green dot denotes mode sample $\hat{\boldsymbol{R}}_k^{\mathbb{C}_{sub}}(j_h)$. Black dot is the mean sample $\bar{\boldsymbol{R}}_k^{\mathbb{C}_{sub}}$. Yellow dots denote foreground samples in the other half of the same cluster.

The final estimations from all clusters indicate "foreground" regions that might contain either the target or distractors, denoted by a set of sample centres $\{\hat{\boldsymbol{X}}_{o,k}(j)\}_{j=1...N_{o,k}}$ where $N_{o,k}$ is the number of observed foreground regions (note that here we use the term "foreground" in a slightly unconventional sense, to refer to both target and target-like distractors).

So far, we have presented a multi-level clustering method with coarse-to-fine sampling to detect target-like regions. The method reduces the computational cost compared to dense sampling over the entire image while improving tracking accuracy compared to methods using a fixed spatial sampling density. The algorithm will next combine the motion history information of both target and distractors, described in section 5.1.2, to definitively associate one foreground region to the target. All remaining foreground regions will be rejected as distractors.

5.1.2 Global dynamic constraint

In highly cluttered scenes, motion cues are important for overcoming the ambiguity caused by appearance similarities between target and distractors [13]. Therefore, we use the motion

history of the tracker and distractors to build a global dynamics model to deterministically associate a single foreground region to the true target.

A. Motion regression model

During camera motion the image coordinates of objects (target or distractors) can abruptly jump to a new image location from one frame to the next. However, the *relative* positions between objects remain comparatively stable. Therefore, these relative positions are used to generate a global motion model. The multi-level clustering procedure (described in the previous section) outputs multiple detected foreground object centres $\{\hat{\boldsymbol{X}}_{o,k}(j)\}_{j=1\ldots N_{o,k}}$ at the kth frame. We now re-write the coordinates of these object centres as:

$$\hat{\boldsymbol{X}}_{o,k}(j) = \bar{\boldsymbol{X}}_{o,k} + \Delta\boldsymbol{X}_{o,k}(j) \tag{5.7}$$

where $\Delta\boldsymbol{X}_{o,k}(j)$ represents the relative displacement between the ith object location and the spatial distribution centre $\bar{\boldsymbol{X}}_{o,k} = \frac{1}{N_{o,k}}\sum_{j=1}^{N_{o,k}}\hat{\boldsymbol{X}}_{o,k}(j)$ of all the object centre estimates.

Over a short time interval, the underlying dynamics of the target can be reasonably approximated by a linear regression model [13]. The relative motion of the target in frame κ is then predicted by a linear regression model: $\Delta\boldsymbol{X}_{t,\kappa} = \beta_0 + \beta_1\kappa + \varepsilon_\kappa$, where β_0, β_1 are the coefficients and $\boldsymbol{\varepsilon}_k$ is a noise term. To estimate the parameters, the algorithm minimises the sum of squared residuals $\sum_{i=1}^{k-1}\varepsilon_\kappa^2$, where $\hat{\beta}_0, \hat{\beta}_1$ is obtained from the historic information of the relative position of the target, by least squares estimates. The predicted relative position of the target center at frame k is:

$$\Delta\hat{\boldsymbol{X}}_{t,k} = \hat{\beta}_0 + \hat{\beta}_1 k \tag{5.8}$$

Note that the relative positions of the target and distractors implicitly encode global information about the scene dynamics.The relative position of foreground object j at the kth frame can be denoted by $\Delta\boldsymbol{X}_{o,k}(j)$, which is computed from equation 5.7. Note that our tracking algorithm is only concerned with solving the *single* target tracking problem, and does not assign or maintain individual IDs for all foreground objects in the scene. Instead, we use the similarity

score of target motion dynamics:

$$w_{D,k}^{j} = e^{-|\Delta \boldsymbol{X}_{o,k}(j) - \Delta \hat{\boldsymbol{X}}_{t,k}|} \tag{5.9}$$

where $w_{D,k}^{j}$ denotes the dynamic similarity score between the predicted target relative position $\Delta \hat{\boldsymbol{X}}_{t,k}$ and the relative position $\Delta \boldsymbol{X}_{o,k}(j)$ of the jth foreground object.

Intuitively, the robustness of this dynamic similarity score, in equation 5.9, corresponds to the complexity and stability of the spatial distribution of the detected foreground objects. If the number of detected foreground objects changes dramatically, it indicates either potential occlusion or newly emerged distractors.

B. Handling dynamic numbers of distractors

For the data association problem, it is crucial to be able to handle situations where the number of detected objects is changing. In such situations, the relative positions can be highly noisy or even invalid because of newly emerged/disappeared objects.

Newly emerged or disappeared foreground objects might be either the target or distractors. Therefore, we use the image coordinates to associate each detected object j with a target-like dynamic matching score $w_{t,k}^{j}$ and distractor-like dynamic matching scores $w_{d,k}^{j,m}$, computed by:

$$\begin{cases} w_{t,k}^{j} = e^{-|\hat{\boldsymbol{X}}_{o,k}(j) - \hat{\boldsymbol{X}}_{t,k-1}|} \\ w_{d,k}^{j,m} = e^{-|\hat{\boldsymbol{X}}_{o,k}(j) - \hat{\boldsymbol{X}}_{d,k-1}^{m}|} \end{cases} \tag{5.10}$$

where $\hat{\boldsymbol{X}}_{t,k-1}, \hat{\boldsymbol{X}}_{d,k-1}^{m}$ are the centers of the target and the m-th distractor in the $k-1$th frame. $\hat{\boldsymbol{X}}_{o,k}(j)$ is the jth detected object at frame k. Here, the exponential function is applied to normalise the likelihood value to occupy the range $(0, 1)$ by equation 5.10. The detected object corresponding to the target should have a high target-like dynamic matching score and also a low distractor-like dynamic matching score, giving a *global* dynamic score $w_{D,k}^{j}$ for the jth

object as:

$$w_{D,k}^{j} = \begin{cases} N_{d,k-1} w_{t,k}^{j} / \sum_{m=1}^{N_{d,k-1}} w_{d,k}^{j,m} \ , \ N_{d,k-1} \neq 0 \\ w_{t,k}^{j} \qquad , \qquad others \end{cases} \tag{5.11}$$

where $N_{d,k-1}$ is the number of distractors in the $k-1$th frame.

C. Target association and occlusion reasoning

The optimal target association (assigning the target to a particular detected object) is confirmed as the one with highest dynamic similarity score $w_{D,k}^{j}$. Moreover, the target-designated candidate region should also have a higher matching likelihood than other (distractor) candidate regions. The final target region is optimally assigned to a candidate region by:

$$\begin{aligned} \hat{j} &= \arg\max \ w_{D,k}^{j} \\ \textbf{s.t.} \ w_{t,k}^{j} &\geq \lambda_d max \ \{w_{d,k}^{j,m}\}_{m=1,\ldots,N_{d,k-1}} \end{aligned} \tag{5.12}$$

where $w_{D,k}^{j}$ is computed from equation 5.9 when the number of detected objects is stable, and from equation 5.11 when the number of detected objects is changing. $N_{d,k-1}$ represents the number of the distractors in the $k-1$th frame while λ_d is a scaling factor in the range 0 to 1. As target-like probability and distractor-like probability are computed from the position, this scaling factor λ_d controls the impact of the position. If no detected object or candidate region satisfies the global dynamic constraint (equation 5.12), then the target is regarded as being in occlusion.

5.2 Experiments

In this section, we first compare our single-target tracker to several other state-of-the-art single-target trackers which are the ones most highly ranked in the most recent state-of-the-art benchmark studies [2, 3, 22]. The compared single target trackers include KCF [9], Struck [25], SCM [7] and CPF [64]. The CT algorithm [8] is considered the most closely related work to

our own and so is also compared.

Additionally, we note that some *multi*-target tracking scenarios (those where many targets have similar appearance) also present interesting challenges for *single*-target tracking, i.e. when one target is selected for tracking, then the other targets become equivalent to severe camouflage and clutter. Therefore, we use test videos from multi-target literature [13, 12] to conduct a second experiment, in which we compare our proposed single-target tracker against multi-target trackers in scenes featuring many identical objects.

In both cases, we evaluate the ability of the trackers to identify a single specific target in each frame, however the multi-target trackers are initialised with significant additional information about all distractor objects and track them as additional targets (enabling them to use data-association to disambiguate the true target). In contrast, our single target tracker is only initialised with information about the single target to be tracked, and must detect and learn about all other objects in the scene on-the-fly.

5.2.1 Single target tracker comparison

A. Dataset

For comparison against other single target trackers, we have selected 15 highly cluttered sequences from state-of-the-art tracking benchmarks [2, 3]. Note that we have specifically chosen not to use the full datasets because: i) these large datasets only contain a few sequences featuring extreme clutter and camouflage, which this work specifically addresses; ii) testing on all sequences introduces confounding factors (non-clutter conditions) making it hard to disambiguate the true capabilities of each algorithm to tackle clutter and camouflage.

To gain a deeper understanding of tracker performance on cluttered scenes, we propose a new set of *sub-attributes* for clutter and camouflage: shape clutter, colour clutter, static camouflage, camera motion-caused camouflage motion, self-moving camouflage. We have *per-frame* annotated all sequences with all sub-attributes. At the time of writing this thesis, this work is still in review for publication. All of the test videos are already publicly available. Once this work becomes accepted for publication, we will publicly release the corresponding set of

Table 5.1: AUC for single target tracking performance in extremely cluttered scenes. Our proposed method significantly outperforms all compared methods on all sub-attributes. (red: best performance; blue: second best performance)

Tracker	Overall	Clutter type		Camouflage motion		
		Colour	Shape	no motion	camera motion	self-motion
Ours	6.4094	6.2877	6.2337	11.9879	6.3740	6.3300
KCF [9]	5.6994	5.5088	5.4689	8.2909	4.6889	5.6541
CPF [64]	5.2830	4.4726	5.0478	10.0545	5.4337	5.1546
SCM [7]	4.4338	4.5308	4.3961	2.7758	3.3757	4.5032
Struck [25]	3.1631	2.5256	3.1031	1.7758	3.0875	3.1748
CT [8]	2.8375	3.0378	2.8246	1.9455	1.8972	2.8559

sub-attribute annotations.

B. Results

For single target tracking, Wu et al. [23] proposed a novel performance metric which uses the area under the curve (AUC) of the overlap rate curve or central pixel distance curve for evaluation. Center-error measure counts the center distance at all frames even after a tracker has failed, which does not reflect the tracker's true performance [154]. Moreover, the centre distance is susceptible to subjective bounding box annotation. Therefore, in this work, we only compare trackers in terms of AUC of the overlap rate curve. We provide the AUC results [2] of each tracker, tested i) over the entire dataset and ii) for frames corresponding to particular sub-attributes in table 5.1. The trade-off overlap rate curve is shown in figure 5.5.

Our proposed tracker outperforms all compared trackers, both overall and also in all sub-attribute categories. The best result is obtained when the camouflage is static. The tracking performance in scenes with both camera motion and camouflage self-motion is quite similar, because our proposed global dynamic constraint exploits the relative positions of target and distractors. In contrast, KCF [9] performs second best for camouflage self-motion, but its performance deteriorates during scenes with camera motion. Since CPF does not assume any target motion model, it performs relatively well (second best) in scenes with camera movement. However, CPF [64] has inferior performance in scenes with colour clutter because it relies solely on a single colour feature for tracking. While SCM [7] and Struck [25] demonstrate excellent overall performance in [2], they still suffer tracking difficulties in handling highly cluttered and

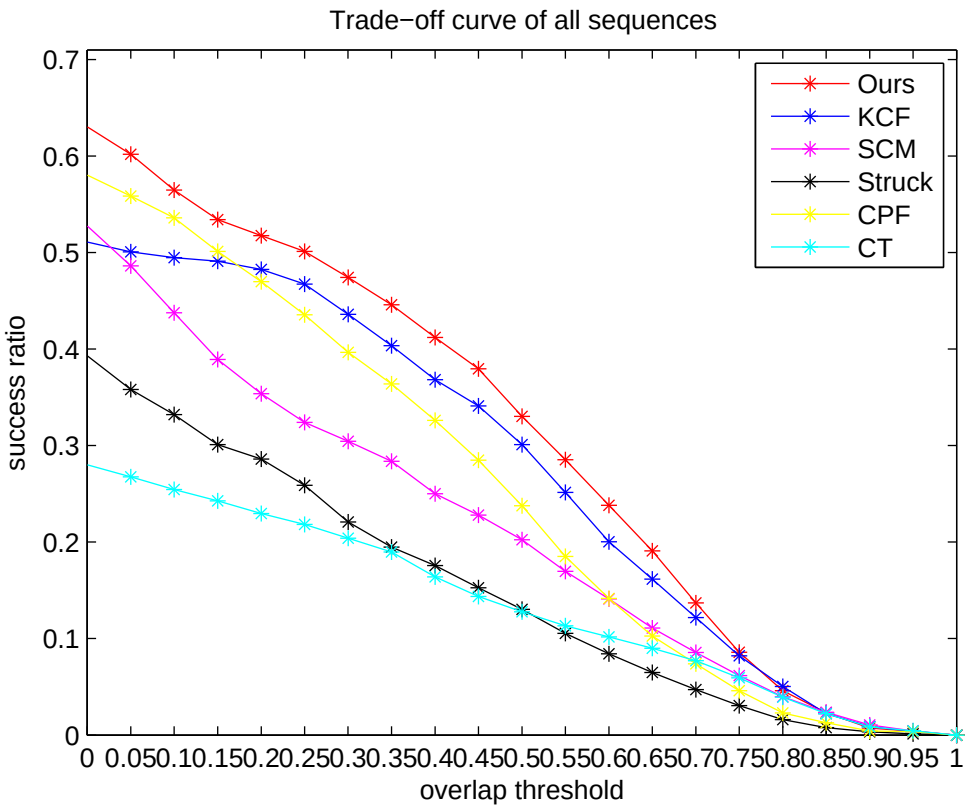

Figure 5.5: The trade-off overlap rate curve of single target trackers.

camouflaged scenes. CT [8] does exploit contextual information, but their algorithm is still based primarily on appearance matching and therefore performs the worst out of the compared methods.

Figure 5.6 illustrates the strong performance of our proposed tracker in extreme clutter and camouflage. Distractors, detected and learned online by our tracker, are indicated by yellow bounding boxes, while the true target is shown with a red bounding box. In frame 139 of sequence *marching*, one distractor shares major overlap with the target, however our proposed multi-level clustering process can still very accurately disambiguate and localise these two objects.

Figure 5.6: Performance of our proposed single target tracker in extremely cluttered and camouflaged scenes. First row: *bolt 1*; Second row: *marching*. Red bounding box: the tracker; Yellow bonding box: adjacent distractors.

5.2.2 Multi-target tracker comparison

It is possible to draw some analogies between our proposed global dynamic constraint (dynamic relationship between target and distractors), and the data association techniques proposed in the multi-target tracking literature. Therefore, we also compare our proposed method against state-of-the-art multi-target trackers [12, 13]. Our proposed single target tracker is initialised with a single bounding box around the true target. In contrast, the multi-target tracker is initialised with multiple bounding boxes (one for each distractor). Therefore the multi-target trackers are given far more a-priori knowledge about the scene and distractors, while our proposed method must detect and learn all distractors on-the-fly.

A. Dataset

We utilise the dataset in [13] which was originally designed for tracking multiple objects with similar appearance. These sequences also present extreme clutter and camouflage challenges for a single-target tracking task, and are therefore well suited for evaluating our proposed algorithm. In addition, we also include one sequence used by TINF [12], which features very similar targets, for a fair comparison. We have modified the ground-truthing on these videos to make them compatible with single-target tracker evaluation methods. The whole dataset contains 14 distinct targets for tracking.

B. Trackers

The multi-target tracking method SMOT [13] is explicitly designed for simultaneously tracking multiple targets which share similar appearance. However, the original algorithm focussed on solving the data association problem, and therefore relied on a-priori knowledge of all ground-truth bounding boxes (i.e. the input to this algorithm is a set of perfect groundtruth detections for all targets in all images of the entire sequence). Therefore, to conduct a meaningful comparison while explicitly highlighting the improvement of our proposed global dynamic constraint, we input the same detections (potential regions that contain the target or distractor) from our proposed multi-level clustering method to SMOT for data association and output the optimized path for the target, computed by SMOT. We also compare our tracker against the state-of-the-art multi-target tracker TINF [12] which extended the original single-target tracker of Struck [25]. For the TINF tracker, we initialise the tracker by providing the ground-truth bounding boxes of all objects (target and distractors) in the first frame, i.e. TINF is allowed significant additional prior knowledge over our own method. For consistency, we still use the same single-target tracking evaluation metric (trade-off curve between overlap threshold and success ratio).

C. Results

The tracking performance of our method versus the two state-of-the-art multi-target trackers is shown in figure 5.7. Clearly, our proposed single-target tracker significantly outperforms the compared methods on this test data. The performance of our method versus SMOT demonstrates the effectiveness of our proposed global dynamic constraint, since the SMOT algorithm has difficulty handling scenes with highly dynamic number of distractors. The strong performance of our method over TINF, suggests that, by properly modelling the context, a single-target tracker can outperform a multi-target tracker, even if the multi-target tracker is initialised with a large amount of additional information about distractors. The TINF method uses SSVM to train a classification model from extracted colour and hog features. However, since TINF did not make full use of motion dynamics, this kind of method performs poorly in highly cluttered scenes with multiple identical objects, since appearance features (colour and HOG) alone are unable to fully disambiguate multiple objects sharing identical appearance. Noticeably, our

proposed tracking method runs at 4.01 fps (using a standard laptop with Matlab), while SMOT has a speed of 1.86 fps. In contrast, since TINF relies on densely sampled feature extraction at every image, the algorithm is significantly slower at around 0.18 fps. Figure 5.8 illustrates the comparative performance of our method and the multi-target trackers in the highly challenging *crowd* sequence.

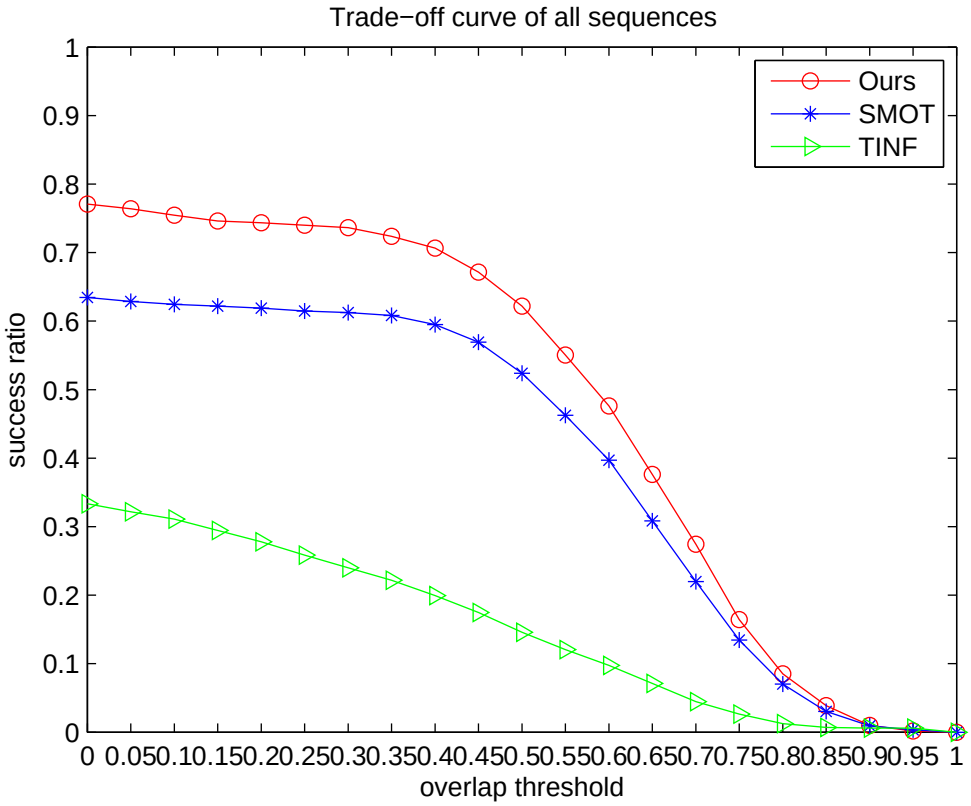

Figure 5.7: Comparative performance of our method (red) versus SMOT (blue) and TINF (green), in terms of ability to consistently track a single target in highly cluttered scenes.

5.3 Summary

This chapter has presented a novel method for tracking a single target in scenes of extreme clutter and camouflage. In contrast to conventional tracking algorithms which only maintain information about the target, the proposed algorithm incorporates a novel multi-level clustering method for online detection and learning of target-like contextual image regions, called distrac-

Figure 5.8: Tracking performance of proposed and compared methods on tracking a single face in the *crowd* sequence. Red: ground truth; yellow: ours; green: SMOT; blue: TINF. Both of the compared methods fail in various frames, with two different failure modes (false positive occlusion detections and erroneous fixation on distractors). In contrast, our proposed method tracks successfully throughout.

tors. To disambiguate the target's path among the distractors, a global dynamic constraint is proposed, which can also detect occlusions when no likely path is found. The proposed method successfully prevents the estimated target location from erroneously jumping to distractors during occlusions or camouflage interactions.

To evaluate our tracker, we have introduced a new set of sub-attributes, e.g. shape clutter, colour clutter, static camouflage, camera motion-caused camouflage motion, self-moving camouflage, and have per-frame annotated a number of public benchmark test sequences with these sub-attributes. Using this dataset,our proposed tracker outperforms all compared trackers, both overall and also in all sub-attribute categories. The best result is obtained when the camouflage is static. The tracking performance in scenes with both camera motion and camouflage self-motion is quite similar, because our proposed global dynamic constraint exploits the relative positions of target and distractors.

In the scenes with camouflage, we have also shown that our single-target tracker outperforms state-of-the-art multi-target trackers, when they are initialised with the bounding boxes of all distractors in each scene. Especially, the performance of our method versus SMOT [13]

demonstrates the effectiveness of our proposed global dynamic constraint. It is also useful to note that the proposed multi-level clustering method and global dynamic constraints could, in principle, be retro-fitted to many other single target trackers to improve their performance in cluttered scenes.

CHAPTER 6

CONCLUSION

In this chapter, we first provide a summary of the thesis in section 6.1. Then, we summarise our contributions in section 6.2 and outline some suggestions for future work in section 6.3.

6.1 Summary

This thesis has investigated single-target tracking problems of arbitrary objects using multi-layered features and contextual information. Without prior information about the target category, the tracker is prone to failure due to many challenges such as significant deformation of the target, occlusion of the target, viewpoint change, illumination variation, background clutter and camouflage. To improve tracking performance under these severe conditions, we aim at: 1) designing a robust tracker in RGB images which could adaptively fuse and update multiple features to overcome a variety of difficult tracking problems, e.g. significant target deformation, rapid and erratic target motion, illumination change and viewpoint change; 2) extending the proposed RGB tracker to handle RGB-D images by showing how depth information can be robustly combined with RGB information during tracking; 3) exploiting the contextual information which could handle camouflage problems in the highly cluttered scenes and providing a deeper analysis of the tracking performance under those severe situations.

6.2 Contributions

The contributions of this thesis can be summarised with respect to three main aspects, listed below:

- In chapter 3, we have presented a new single target tracker incorporating a multi-level appearance model. An adaptive clustered decision tree method has been proposed to select the minimum combination of features necessary to sufficiently represent each target part at each frame, thereby providing robustness with computational efficiency. Based on two different evaluation protocols, we have tested the tracker using two different tracking benchmarks [2, 3] comprising a total of 70 sequences and comparing our method against more than 50 other tracking algorithms from the literature. The proposed tracker was demonstrated to be significantly more robust than the state-of-the-art methods, while also offering competitive tracking precision.

- In chapter 4, we proposed a single target tracking algorithm in RGB-D images, incorporating a novel adaptive multi-feature local-global target model with both temporal and spatial constraints. At the bounding box level, temporal consistency was used to adaptively fuse information from both RGB and depth images to find a candidate target region. In frames where one or more features are temporally inconsistent, this global candidate region was further split into local candidate regions for matching to target parts. We derived a physical constraint from the depth context, which robustly accommodated large target depth variation, while still enabling accurate reasoning about occlusions. For evaluation, we introduced a new RGB-D benchmark dataset with per-frame annotated attributes and extensive bias analysis. Our tracker has been evaluated using two different methodologies [2, 3] and in both cases it outperformed other state-of-the-art RGB-D trackers.

- In chapter 5, we have shown how distracting contextual regions can be detected, learned and explicitly exploited to support a single target tracker under conditions of extreme clutter and camouflage, including frequent occlusions by objects with similar appearance to the target. A multi-level clustering method has been proposed for online detection

and learning of multiple target-like regions, called distractors, when they appeared near to the true target. To distinguish the target from these distractors, we used global dynamic constraints (derived from target and distractor information) to optimize the estimated target trajectory. To evaluate our tracker, we have augmented publicly available benchmark videos, by proposing a new set of clutter sub-attributes, and annotating these sub-attributes for all frames in all sequences. Using this dataset, we have shown that our proposed method outperforms other state-of-the-art single target trackers on highly cluttered scenes.

6.3 Future work

There are some interesting works related to tracking remaining to be done. The trade-off between tracking robustness and accuracy is an important problem. While our proposed trackers have achieved state-of-the-art robustness, the tracking accuracy could be further improved. Since we applied a segmentation method for constructing the local parts (modelled as superpixels), those constructed local parts contain homogeneous pixels which is good for robustness. However, such homogeneous regions omit certain kinds of saliency (e.g. edges, corners or edge junctions) of the local parts which can be helpful for accurate localisation. Therefore, one possible research direction for the current tracker is to design another type of local parts which are be more discriminative, such as SIFT regions [89]. A new algorithm might combine homogeneous local parts, which aid robustness, with more other kinds of local parts which are better for aiding accuracy.

The current datasets were constructed for testing the trackers described in this thesis. To avoid the overfitting problem, we will enlarge the size of this test dataset. Additionally, we will also construct the training dataset with the same quality, which can achieve a more fair comparison by: 1) separately comparing the algorithms with and without a training stage; 2) defining a common training set to ensure that all the training-based methods have the same prior information.

Even though we have proposed several tracking algorithms which could advance the current state-of-the-art tracking research, visual object tracking remains an open and active research area. This is because, with one initialised bounding box, visual object tracking is an ill-posed

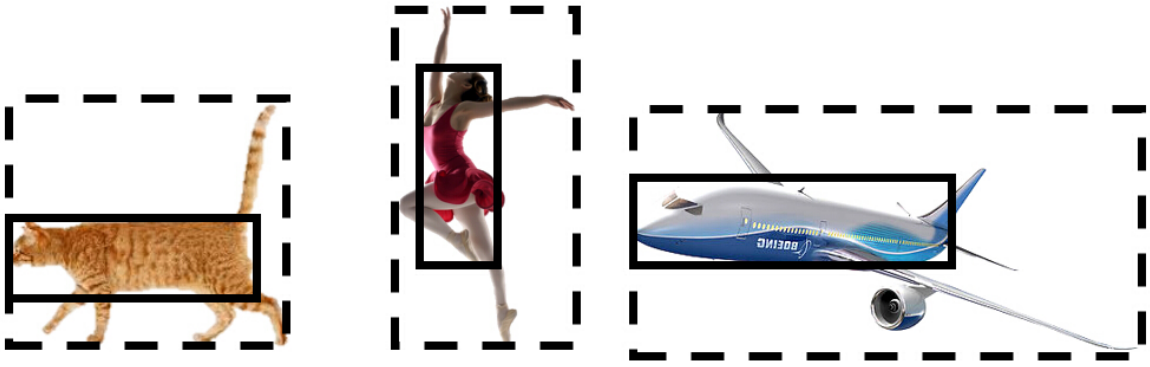

Figure 6.1: The ambiguity of defining the ground truth of arbitrary objects. Bounding boxes with dotted lines represent the ground truth defined in terms of the smallest possible bounding box which completely encloses the entire tracked object. Bounding boxes with solid lines use an intuitive human notion of "the main part" of an object to define the ground truth.

problem since the bounding box used for initialisation will almost always contain some background pixels and the tracker has no way of knowing what it should track inside the bounding box. Firstly, this causes a ground truth ambiguity [22], shown in figure 6.1, which leads to difficulties defining the boundary of the target object and causes evaluation difficulties. A larger bounding box contains more information about the target, but also includes many non-target background pixels. In contrast, a smaller bounding box contains less background pixels but also loses some target information. Secondly, when the bounding box contains more than one entity, it also causes an ambiguity in tracking objectives (what we should really track), shown in figure 6.2. In figure 6.2, without any prior information about the target category, it is not clear whether the tracker should track the glasses or eyes. In this situation, what would it mean to perform tracking "correctly" for a video in which the boy removes his glasses, puts them down, and walks away?

Since the current trackers have no prior information about target category, they are often built based on some heuristic assumptions, e.g. motion smoothness, structure consistency and colour invariance. With the help of such heuristic assumptions, the tracking performance is

Figure 6.2: An example of ambiguity in tracking objectives (picture is obtained from [157]). The red bounding box is provided as the ground truth in the first frame. It is not clear whether the tracker should track the glasses or eyes.

often improved where those assumptions hold true. However, such assumptions also interfere with performance when the object changes or moves in a way that is different from what was assumed. Therefore, an interesting research direction in the future is whether the features used for tracking could go beyond the pixel level and incorporate some common-sense knowledge, e.g. physics, intentionality, causality and functionality, which avoids the need for such heuristics.

REFERENCES

[1] Jingjing Xiao, Rustam Stolkin, and Aleš Leonardis. Single target tracking using adaptive clustered decision trees and dynamic multi-level appearance models. In *CVPR*, 2015.

[2] VTB benchmark dataset. https://sites.google.com/site/trackerbenchmark/benchmarks/v10.

[3] The VOT chanllenge. http://www.votchallenge.net/.

[4] A Smeulders, D Chu, Rita Cucchiara, Simone Calderara, Afshin Dehghan, and Mubarak Shah. Visual tracking: An experimental survey. *PAMI*, pages 1442 – 1468, 2014.

[5] Princeton tracking benchmark. http://tracking.cs.princeton.edu/index.html.

[6] Luka Čehovin, Matej Kristan, and Aleš Leonardis. Robust visual tracking using an adaptive coupled-layer visual model. *PAMI*, pages 941 – 953, Apr 2013.

[7] Wei Zhong, Huchuan Lu, and Ming-Hsuan Yang. Robust object tracking via sparsity-based collaborative model. In *CVPR*, pages 1838–1845. IEEE, 2012.

[8] Thang Ba Dinh, Nam Vo, and Gérard Medioni. Context tracker: Exploring supporters and distracters in unconstrained environments. In *CVPR*, pages 1177–1184. IEEE, 2011.

[9] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *PAMI*, 37(3):583–596, 2015.

[10] Massimo Camplani, Sion Hannuna, Majid Mirmehdi, Dima Damen, Adeline Paiement, Lili Tao, and Tilo Burghardt. RGB-D tracking: Depth scaling kernelised correlation filters DS-KCF. In *BMVC*. Springer, 2015.

[11] Kourosh Meshgia, Shin-ichi Maedaa, Shigeyuki Obaa, Henrik Skibbea, Yu-zhe Lia, and Shin Ishiia. Occlusion aware particle filter tracker to handle complex and persistent occlusions. *CVIU*, 2015.

[12] Afshin Dehghan, Yicong Tian, Philip HS Torr, and Mubarak Shah. Target identity-aware network flow for online multiple target tracking. In *CVPR*, pages 1146–1154, 2015.

[13] Caglayan Dicle, Octavia I Camps, and Mario Sznaier. The way they move: Tracking multiple targets with similar appearance. In *ICCV*, pages 2304–2311. IEEE, 2013.

[14] Xiaogang Wang. Intelligent multi-camera video surveillance: A review. *Pattern recognition letters*, 34(1):3–19, 2013.

[15] Naiyan Wang, Jianping Shi, Dit-Yan Yeung, and Jiaya Jia. Understanding and diagnosing visual tracking systems. *ICCV*, 2015.

[16] Karel Lebeda, Simon Hadfield, and Richard Bowden. Exploring causal relationships in visual object tracking. In *ICCV*, 2015.

[17] Tianzhu Zhang, Si Liu, Changsheng Xu, Shuicheng Yan, Bernard Ghanem, Narendra Ahuja, and Ming-Hsuan Yang. Structural sparse tracking. In *CVPR*, pages 150–158, June 2015.

[18] Liujuan Cao and Rongrong Ji. Robust depth-based object tracking from a moving binocular camera. *Signal Processing*, pages 154–161, 2015.

[19] Yan Chen, Yingju Shen, Xin Liu, and Bineng Zhong. 3D object tracking via image sets and depth-based occlusion detection. *Signal Processing*, pages 146–153, 2015.

[20] Shuran Song and Jianxiong Xiao. Tracking revisited using RGBD camera: Unified benchmark and baselines. In *ICCV*, pages 233–240, 2013.

[21] M. Kristan et al. The visual object tracking vot2013 challenge results. In *Computer Vision Workshops (ICCVW), 2013 IEEE International Conference on*, pages 98–111, Dec 2013.

[22] Annan Li, Min Lin, Yi Wu, Ming-Hsuan Yang, and Shuicheng Yan. Nus-pro: A new visual tracking challenge. *PAMI*, PP(99):1–1, 2015.

[23] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *CVPR*, pages 2411–2418. IEEE, 2013.

[24] Hanxuan Yang, Ling Shao, Feng Zheng, Liang Wang, and Zhan Song. Recent advances and trends in visual tracking: A review. *Neurocomputing*, 74(18):3823–3831, 2011.

[25] Sam Hare, Amir Saffari, and Philip HS Torr. Struck: Structured output tracking with kernels. In *ICCV*, pages 263–270. IEEE, 2011.

[26] Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. Recurrent network models for human dynamics. ICCV, 2015.

[27] Xianbin Cao, Changcheng Gao, Jinhe Lan, Yuan Yuan, and Pingkun Yan. Ego motion guided particle filter for vehicle tracking in airborne videos. *Neurocomputing*, 124:168 – 177, 2014.

[28] Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. Training a feedback loop for hand pose estimation. In *ICCV*, 2015.

[29] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *ACM computing surveys*, 38(4):13, 2006.

[30] V Salari and Ishwar K. Sethi. Feature point correspondence in the presence of occlusion. *PAMI*, (1):87–91, 1990.

[31] Cor J Veenman, Marcel JT Reinders, and Eric Backer. Resolving motion correspondence for densely moving points. *PAMI*, 23(1):54–72, 2001.

[32] Jeffrey F Cohn, Adena J Zlochower, James J Lien, and Takeo Kanade. Feature-point tracking by optical flow discriminates subtle differences in facial expression. In *FG*, pages 396–401. IEEE, 1998.

[33] Prithiraj Tissainayagam and David Suter. Object tracking in image sequences using point features. *Pattern Recognition*, 38(1):105–113, 2005.

[34] Carlo Tomasi and Takeo Kanade. *Detection and tracking of point features*. School of Computer Science, Carnegie Mellon Univ. Pittsburgh, 1991.

[35] Marcelo Bertalmio, Guillermo Sapiro, and Gregory Randall. Morphing active contours. *PAMI*, 22(7):733–737, 2000.

[36] Michael Isard and Andrew Blake. Condensationconditional density propagation for visual tracking. *International journal of computer vision*, 29(1):5–28, 1998.

[37] Frederic Leymarie and Martin D Levine. Tracking deformable objects in the plane using an active contour model. *PAMI*, 15(6):617–634, 1993.

[38] Michael Isard and Andrew Blake. Contour tracking by stochastic propagation of conditional density. In *ECCV*, pages 343–356. Springer, 1996.

[39] Weiming Hu, Xue Zhou, Wei Li, Wenhan Luo, Xiaoqin Zhang, and Steve Maybank. Active contour-based visual tracking by integrating colors, shapes, and motions. *TIP*, 22(5):1778–1792, 2013.

[40] Han-Ul Kim, Dae-Youn Lee, Jae-Young Sim, and Chang-Su Kim. Sowp: Spatially ordered and weighted patch descriptor for visual tracking. 2015.

[41] Jianbo Shi and Carlo Tomasi. Good features to track. In *CVPR*, pages 593–600. IEEE, 1994.

[42] Hai Tao, Harpreet S Sawhney, and Rakesh Kumar. Object tracking with bayesian estimation of dynamic layer representations. *PAMI*, 24(1):75–89, 2002.

[43] Michael J Black and Allan D Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision*, 26(1):63–84, 1998.

[44] Shai Avidan. Support vector tracking. *PAMI*, 26(8):1064–1072, 2004.

[45] Xiaogang Wang, Gianfranco Doretto, Thomas Sebastian, Jens Rittscher, and Peter Tu. Shape and appearance context modeling. In *ICCV*, pages 1–8. IEEE, 2007.

[46] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, volume 1, pages I–511. IEEE, 2001.

[47] Constantine P Papageorgiou, Michael Oren, and Tomaso Poggio. A general framework for object detection. In *ICCV*, pages 555–562. IEEE, 1998.

[48] Sri-Kaushik Pavani, David Delgado, and Alejandro F Frangi. Haar-like features with optimally weighted rectangles for rapid object detection. *Pattern Recognition*, 43(1):160–172, 2010.

[49] Timo Ojala, Matti Pietikäinen, and David Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern recognition*, 29(1):51–59, 1996.

[50] Toan Thanh Do, Khiem Ngoc Doan, Thai Hoang Le, and Bac Hoai Le. Boosted of haar-like features and local binary pattern based face detection. In *International Conference on Computing and Communication Technologies*, pages 1–8. IEEE, 2009.

[51] Dong-Chen He and Li Wang. Texture features based on texture spectrum. *Pattern Recognition*, 24(5):391–399, 1991.

[52] Guoying Zhao, Timo Ahonen, Jiří Matas, and Matti Pietikäinen. Rotation-invariant image and video description with local binary pattern features. *TIP*, 21(4):1465–1477, 2012.

[53] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, volume 1, pages 886–893. IEEE, 2005.

[54] Anna Bosch, Xavier Muñoz, and Robert Martí. Which is the best way to organize/classify images by content? *Image and vision computing*, 25(6):778–791, 2007.

[55] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y Ng. Efficient sparse coding algorithms. In *NIPS*, pages 801–808, 2006.

[56] Tiezheng Ge, Qifa Ke, and Jian Sun. Sparse-coded features for image retrieval. In *BMVC*, volume 6, 2013.

[57] Meng Yang, Dengxin Dai, Linlin Shen, and Luc Van Gool. Latent dictionary learning for sparse representation based classification. In *CVPR*, pages 4138–4145. IEEE, 2014.

[58] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online dictionary learning for sparse coding. In *ICML*, pages 689–696. ACM, 2009.

[59] Ali S Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *CVPRW*, pages 512–519. IEEE, 2014.

[60] Steve Lawrence, C Lee Giles, Ah Chung Tsoi, and Andrew D Back. Face recognition: A convolutional neural-network approach. *Neural Networks, IEEE Transactions on*, 8(1):98–113, 1997.

[61] Robert T Collins, Yanxi Liu, and Marius Leordeanu. Online selection of discriminative tracking features. *PAMI*, 27(10):1631–1643, 2005.

[62] Songtao Wu, Yuesheng Zhu, and Qing Zhang. A new robust visual tracking algorithm based on transfer adaptive boosting. *Mathematical Methods in the Applied Sciences*, 35(17):2133–2140, 2012.

[63] Iain Matthews, Takahiro Ishikawa, and Simon Baker. The template update problem. *PAMI*, 26(6):810–815, 2004.

[64] Katja Nummiaro, Esther Koller-Meier, and Luc Van Gool. An adaptive color-based particle filter. *Image and vision computing*, 21(1):99–110, 2003.

[65] François Ennesser and Gérard Medioni. Finding waldo, or focus of attention using local color information. *PAMI*, 17(8):805–809, 1995.

[66] Rustam Stolkin, Ionut Florescu, and George Kamberov. An adaptive background model for camshift tracking with a moving camera. In *6th International Conference on Advances in Pattern Recognition*, pages 147–151. Citeseer, 2007.

[67] Junseok Kwon and Kyoung Mu Lee. Highly nonrigid object tracking via patch-based dynamic appearance modeling. *PAMI*, 35(10):2427–2441, 2013.

[68] Xu Jia, Huchuan Lu, and Ming-Hsuan Yang. Visual tracking via adaptive structural local sparse appearance model. In *CVPR*, pages 1822–1829. IEEE, 2012.

[69] Amit Adam, Ehud Rivlin, and Ilan Shimshoni. Robust fragments-based tracking using the integral histogram. In *CVPR*, volume 1, pages 798–805. IEEE, 2006.

[70] Shu Wang, Huchuan Lu, Fan Yang, and Ming-Hsuan Yang. Superpixel tracking. In *ICCV*, pages 1323–1330. IEEE, 2011.

[71] Chao Ma, Jia-Bin Huang, Xiaokang Yang, and Ming-Hsuan Yang. Hierarchical convolutional features for visual tracking. In *ICCV*, 2015.

[72] Lijun Wang, Wanli Ouyang, Xiaogang Wang, and Huchuan Lu. Visual tracking with fully convolutional networks. In *ICCV*, 2015.

[73] Tianzhu Zhang, Si Liu, Changsheng Xu, Shuicheng Yan, Bernard Ghanem, Narendra Ahuja, and Ming-Hsuan Yang. Structural sparse tracking. In *CVPR*, pages 150–158, 2015.

[74] Yang Lu, Tianfu Wu, and Song-Chun Zhu. Online object tracking, learning and parsing with and-or graphs. In *CVPR*, pages 3462 – 3469. IEEE, 2014.

[75] Seunghoon Hong and Bohyung Han. Visual tracking by sampling tree-structured graphical models. In *ECCV*, pages 1–16. Springer, 2014.

[76] Anil Kumar Bhattacharya On a measure of divergence between two statistical populations defined by their probability distributions. In *Bulletin of the Calcutta Mathematical Society, issue 35*, pages 99–110, 1943.

[77] Anil Bhattacharyya. On a measure of divergence between two multinomial populations. *Sankhyā: The Indian Journal of Statistics*, pages 401–406, 1946.

[78] Per-Erik Danielsson. Euclidean distance mapping. *Computer Graphics and Image Processing*, 14(3):227–248, 1980.

[79] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *IJCV*, 40(2):99–121, 2000.

[80] Zhe Chen, Zhibin Hong, and Dacheng Tao. An experimental survey on correlation filter-based tracking. *arXiv preprint arXiv:1509.05520*, 2015.

[81] David S Bolme, Bruce Draper, J Ross Beveridge, et al. Average of synthetic exact filters. In *CVPR*, pages 2105–2112. IEEE, 2009.

[82] David G Kleinbaum and Mitchel Klein. *Analysis of Matched Data Using Logistic Regression*. Springer, 2010.

[83] Yoav Freund, Robert Schapire, and N Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

[84] Helmut Grabner, Christian Leistner, and Horst Bischof. Semi-supervised on-line boosting for robust tracking. In *ECCV*, pages 234–247. Springer, 2008.

[85] Toufiq Parag, Fatih Porikli, and Ahmed Elgammal. Boosting adaptive linear weak classifiers for online learning and tracking. In *CVPR*, pages 1–8. IEEE, 2008.

[86] Zhuowen Tu. Probabilistic boosting-tree: Learning discriminative models for classification, recognition, and clustering. In *ICCV*, volume 2, pages 1589–1596. IEEE, 2005.

[87] Christopher JC Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[88] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Kernel-based object tracking. *PAMI*, 25(5):564–577, 2003.

[89] Federico Pernici and Alberto Del Bimbo. Object tracking by oversampling local features. *PAMI*, 36:2538 – 2551, 2013.

[90] Rustam Stolkin and Mohammed Talha. Particle filter tracking of camouflaged targets by adaptive fusion of thermal and visible spectra camera data. *IEEE Sensors*, 2014.

[91] Rustam Stolkin, David Rees, Mohammed Talha, and Ionut Florescu. Bayesian fusion of thermal and visible spectra camera data for region based tracking with rapid background adaptation. In *IEEE Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 192–199. IEEE, 2012.

[92] Allan D Jepson, David J Fleet, and Thomas F El-Maraghi. Robust online appearance models for visual tracking. *PAMI*, 25(10):1296–1311, 2003.

[93] Shaohua Kevin Zhou, Rama Chellappa, and Baback Moghaddam. Visual tracking and recognition using appearance-adaptive models in particle filters. *TIP*, 13(11):1491–1506, 2004.

[94] Yi Wu, Jian Cheng, Jinqiao Wang, and Hanqing Lu. Real-time visual tracking via incremental covariance tensor learning. In *ICCV*, pages 1631–1638. IEEE, 2009.

[95] Xi Li, Weiming Hu, Zhongfei Zhang, Xiaoqin Zhang, Mingliang Zhu, and Jian Cheng. Visual tracking via incremental log-euclidean riemannian subspace learning. In *CVPR*, pages 1–8. IEEE, 2008.

[96] Xi Li, Weiming Hu, Zhongfei Zhang, Xiaoqin Zhang, and Guan Luo. Robust visual tracking based on incremental tensor subspace learning. In *ICCV*, pages 1–8. IEEE, 2007.

[97] Qing Wang, Feng Chen, Wenli Xu, and Ming-Hsuan Yang. Online discriminative object tracking with local sparse representation. In *WACV*, pages 425–432. IEEE, 2012.

[98] Yao Sui, Yafei Tang, and Li Zhang. Discriminative low-rank tracking. 2015.

[99] Jianyu Wang, Xilin Chen, and Wen Gao. Online selecting discriminative tracking features using particle filter. In *CVPR*, volume 2, pages 1037–1042. IEEE, 2005.

[100] Xiaoqin Zhang, Weiming Hu, Steve Maybank, and Xi Li. Graph based discriminative learning for robust and efficient object tracking. In *ICCV*, pages 1–8. IEEE, 2007.

[101] Geoffrey E Hinton. Products of experts. In *ICANN*, volume 1, pages 1–6. IET, 1999.

[102] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. Robust object tracking with online multiple instance learning. *PAMI*, 33(8):1619–1632, 2011.

[103] Helmut Grabner and Horst Bischof. On-line boosting and vision. In *CVPR*, volume 1, pages 260–267. IEEE, 2006.

[104] Georg Nebehay and Roman Pflugfelder. Clustering of static-adaptive correspondences for deformable object tracking. In *CVPR*, 2015.

[105] Yang Li, Jianke Zhu, and Steven CH Hoi. Reliable patch trackers: Robust visual tracking by exploiting reliable patches. In *CVPR*, pages 353–361, 2015.

[106] Ting Liu, Gang Wang, and Qingxiong Yang. Real-time part-based visual tracking via adaptive correlation filters. *CVPR*, 2015.

[107] Ming Tang and Jiayi Feng. Multi-kernel correlation filter for visual tracking. 2015.

[108] Chao Ma, Xiaokang Yang, Chongyang Zhang, and Ming-Hsuan Yang. Long-term correlation tracking. In *CVPR*, pages 5388–5396, 2015.

[109] David A Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. Incremental learning for robust visual tracking. *IJCV*, 77(1-3):125–141, 2008.

[110] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-learning-detection. *PAMI*, 34(7):1409–1422, 2012.

[111] Horst Possegger, Thomas Mauthner, and Horst Bischof. In defense of color-based model-free tracking. In *CVPR*, pages 2113–2120, 2015.

[112] Xue Mei and Haibin Ling. Robust visual tracking using l1 minimization. In *ICCV*, pages 1436–1443. IEEE, 2009.

[113] Jakub Segen and Senthil Kumar. Shadow gestures: 3d hand pose estimation using a single camera. In *CVPR*, volume 1. IEEE, 1999.

[114] Iason Oikonomidis, Nikolaos Kyriazis, and Antonis A Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In *BMVC*, volume 1, page 3, 2011.

[115] Omid Hosseini Jafari, Dennis Mitzel, and Bastian Leibe. Real-time rgb-d based people detection and tracking for mobile robots and head-worn cameras. In *ICRA*, pages 5636–5643. IEEE, 2014.

[116] Sijin Li, Weichen Zhang, and Antoni B Chan. Maximum-margin structured learning with deep networks for 3d human pose estimation. *ICCV*, 2015.

[117] Gregory P. Meyer, Shalini Gupta, Iuri Frosio, Dikpal Reddy, and Kautz Jan. Robust model-based 3d head pose estimation. *ICCV*, 2015.

[118] H. Grabner, J. Matas, L. Van Gool, and P. Cattin. Tracking the invisible: Learning where the object might be. In *CVPR*, pages 1285–1292, June 2010.

[119] Ming Yang, Ying Wu, and Gang Hua. Context-aware visual tracking. *PAMI*, 31(7):1195–1209, 2009.

[120] Jaco Vermaak, Arnaud Doucet, and Patrick Pérez. Maintaining multimodality through mixture tracking. In *ICCV*, pages 1110–1116. IEEE, 2003.

[121] Kenji Okuma, Ali Taleghani, Nando De Freitas, James J Little, and David G Lowe. A boosted particle filter: Multitarget detection and tracking. In *ECCV*, pages 28–39. Springer, 2004.

[122] Changjiang Yang, Ramani Duraiswami, and Larry Davis. Fast multiple object tracking via a hierarchical particle filter. In *ICCV*, volume 1, pages 212–219. IEEE, 2005.

[123] Jerome Berclaz, Francois Fleuret, Engin Turetken, and Pascal Fua. Multiple object tracking using k-shortest paths optimization. *PAMI*, 33(9):1806–1819, 2011.

[124] Horesh Ben Shitrit, Jerome Berclaz, Francois Fleuret, and Pascal Fua. Tracking multiple people under global appearance constraints. In *ICCV*, pages 137–144. IEEE, 2011.

[125] Sheng Chen, Alan Fern, and Sinisa Todorovic. Multi-object tracking via constrained sequential labeling. In *CVPR*, pages 1130–1137. IEEE, 2014.

[126] Jingjing Xiao, Rustam Stolkin, and Ales Leonardis. Multi-target tracking in team-sports videos via multi-level context-conditioned latent behaviour models. In *BMVC*, 2014.

[127] Oliver Zendel, Markus Murschitz, Martin Humenberger, and Wolfgang Herzner. Cv-hazop: Introducing test data validation for computer vision. June 2015.

[128] Arnold WM Smeulders, Dung M Chu, Rita Cucchiara, Simone Calderara, Afshin Dehghan, and Mubarak Shah. Visual tracking: an experimental survey. *PAMI*, 36(7):1442–1468, 2014.

[129] Yaowu Wu, Jungyoul Lim, and Ming-Hsuan Yang. Object tracking benchmark. *PAMI*, 2015.

[130] Matej Kristan, Jiri Matas, Ales Leonardis, Tomas Vojir, Roman Pflugfelder, Gustavo Fernandez, Georg Nebehay, Fatih Porikli, and Luka Cehovin. A novel performance evaluation methodology for single-target trackers. *PAMI*, 2015.

[131] Antonio Torralba, Alexei Efros, et al. Unbiased look at dataset bias. In *CVPR*, pages 1521–1528. IEEE, 2011.

[132] Jean Ponce, Tamara L Berg, Mark Everingham, David A Forsyth, Martial Hebert, Svetlana Lazebnik, Marcin Marszalek, Cordelia Schmid, Bryan C Russell, Antonio Torralba, et al. Dataset issues in object recognition. In *Toward category-level object recognition*, pages 29–48. Springer, 2006.

[133] Aditya Khosla, Tinghui Zhou, Tomasz Malisiewicz, Alexei A Efros, and Antonio Torralba. Undoing the damage of dataset bias. In *ECCV*, pages 158–171. Springer, 2012.

[134] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, pages 79–86, 1951.

[135] Haibin Ling and Kazunori Okada. Diffusion distance for histogram comparison. In *CVPR*, volume 1, pages 246–253. IEEE, 2006.

[136] Jingjing Xiao, Rustam Stolkin, and Aleš Leonardis. An enhanced adaptive coupled-layer LGTracker++. In *ICCV visual object tracking workshop*, volume 2, page 5. 2013.

[137] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Susstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *PAMI*, 34(11):2274–2282, 2012.

[138] Cher Keng Heng, Sumio Yokomitsu, Yuichi Matsumoto, and Hajime Tamura. Shrink boost for selecting multi-lbp histogram features in object detection. In *CVPR*, pages 3250–3257. IEEE, 2012.

[139] Michael Felsberg. Enhanced distribution field tracking using channel representations. In *ICCV visual object tracking workshop*, pages 121–128. IEEE, 2013.

[140] Tomáš Vojíř and Jiří Matas. Robustifying the flock of trackers. In *16th Computer Vision Winter Workshop*, pages 91–97, 2011.

[141] Zhaowei Cai, Longyin Wen, Jianwei Yang, Zhen Lei, and Stan Z Li. Structured visual tracking with dynamic graph. In *ACCV*, pages 86–97. Springer, 2013.

[142] Martin Danelljan, Gustav Häger, Fahad Shahbaz Khan, and Michael Felsberg. Accurate scale estimation for robust visual tracking. In *BMVC*, 2014.

[143] Yang Li and Jianke Zhu. A scale adaptive kernel correlation filter tracker with feature integration. In *ECCV visual object tracking workshop*, pages 254–265. 2014.

[144] Joao F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. Exploiting the circulant structure of tracking-by-detection with kernels. In *ECCV*, pages 702–715. Springer, 2012.

[145] Joost Van De Weijer, Cordelia Schmid, Jakob Verbeek, and Diane Larlus. Learning color names for real-world applications. *TIP*, 18(7):1512–1523, 2009.

[146] Martin Danelljan, Fahad Shahbaz Khan, Michael Felsberg, and Joost van de Weijer. Adaptive color attributes for real-time visual tracking. In *CVPR*, pages 1090–1097. IEEE, 2014.

[147] David S Bolme, J Ross Beveridge, Bruce Draper, Yui Man Lui, et al. Visual object tracking using adaptive correlation filters. In *CVPR*, pages 2544–2550. IEEE, 2010.

[148] Tomas Vojir and Jiri Matas. Online adaptive hidden markov model for multi-tracker fusion. In *arXiv.org*, 2015.

[149] Eric W Weisstein. Heaviside step function. 2002.

[150] Depth sensors comparison. http://wiki.ipisoft.com/Depth_Sensors_Comparison.

[151] Gerald Rauscher, Daniel Dube, and Andreas Zell. A comparison of 3d sensors for wheeled mobile robots. In *Intelligent Autonomous Systems*, pages 29–41. Springer, 2016.

[152] Syed Muhammad Abbas and Abubakr Muhammad. Outdoor rgb-d slam performance in slow mine detection. In *Robotics; Proceedings of ROBOTIK 2012; 7th German Conference on*, pages 1–6. VDE, 2012.

[153] Openni programmers guide. http://com.occipital.openni.s3.amazonaws.com/OpenNI_Programmers_Guide.pdf.

[154] Luka Čehovin, Matej Kristan, and Aleš Leonardis. Is my new tracker really better than yours? In *WACV*, pages 540 – 547. IEEE, IEEE, Mar 2014.

[155] In Jae Myung. Tutorial on maximum likelihood estimation. *Journal of mathematical Psychology*, 47(1):90–100, 2003.

[156] Jingjing Xiao and Mourad Oussalah. Collaborative tracking for multiple objects in the presence of inter-occlusions. *TCSVT*, 2015.

[157] Image source. http://www.eye0572.com/peijing/429.html.